Music Production Behaviour Modelling

Joseph T Colonel

PhD thesis

School of Electronic Engineering and Computer Science
Queen Mary University of London

2022

Abstract

The new millennium has seen an explosion of computational approaches to the study of music production, due in part to the decreasing cost of computation and the increase of digital music production techniques. The rise of digital recording equipment, MIDI, digital audio workstations (DAWs), and software plugins for audio effects led to the digital capture of various processes in music production. This discretization of traditionally analogue methods allowed for the development of intelligent music production, which uses machine learning to numerically characterize and automate portions of the music production process. One algorithm from the field referred to as "reverse engineering a multitrack mix" can recover the audio effects processing used to transform a multitrack recording into a mixdown in the absence of information about how the mixdown was achieved.

This thesis improves on this method of reverse engineering a mix by leveraging recent advancements in machine learning for audio. Using the differentiable digital signal processing paradigm, greybox modules for gain, panning, equalisation, artificial reverberation, memoryless waveshaping distortion, and dynamic range compression are presented. These modules are then connected in a mixing chain and are optimized to learn the effects used in a given mixdown.

Both objective and perceptual metrics are presented to measure the performance of these various modules in isolation and within a full mixing chain. Ultimately a fully differentiable mixing chain is presented that outperforms previously proposed methods to reverse engineer a mix. Directions for future work are proposed to improve characterization of multitrack mixing behaviours.

Acknowledgements

First and foremost I would like to thank my advisor Joshua Reiss for his guidance, patience, and mentorship throughout this PhD. I could have never imagined the direction my life would take after our brief conversation at DAFx 2018, and I am grateful for the opportunities I have been given to pursue my twin passions of music and machine learning. I am also incredibly grateful to my second supervisor George Fazekas and independent assessor Mathieu Barthet for guiding my work and providing feedback. A special thank you to Mathieu who graciously collaborated with me to produce student mixes and comments in his 2019 sound production and recording techniques course.

I would also like to thank Kazunobu Kondo and Yu Takahashi for their continual support throughout the PhD. Our bi-monthly meetings were instrumental in assessing the work and helping me get across the finish line.

A huge thank you to the Yamaha Corporation for funding this PhD and proposing the research topic. Exploring mix engineering and machine learning has broadened my horizons and deepened my love for music beyond what I could have ever anticipated.

To Sam Keene, for your constant trans-Atlantic mentorship and friendship, thank you.

To Christian, Marco, and Ilias, thank you for your collaboration and your friendship. I can't wait to see where your research leads you.

To Silvia, Lee, Steve, and Kat, it's an incredibly rare thing to meet your heroes, and even rarer to get to call them your friends.

To Sig, Lina, and Jean, I could not have asked for a better group of people to survive lockdown with. And I apologize for cooking nothing but curry for months on end.

To Benny and Hadeel, for welcoming me into your home and hearts whenever I needed, and for letting me nap on your couch mid-conversation, thank you.

To Ines, Dario, and Duarte, the chaos house brought nonstop joy, excitement, and love to my time in London. A thousand kisses.

To V, for putting up with my nonstop ribbing, for indulging my absurd taste in anime, and for keeping me steady when I wanted to fall apart, I can never repay you.

To Aggela, I've learned everything I know about mix engineering from you. You're an amazing mentor, incredibly driven, and smarter than you give yourself credit for. Thank you for everything.

To Courtney, my day one friend at QMUL, my guide to living in London as an American, my July 4 BBQ buddy, my Friendsgiving co-chef, my Slush Puppie hype man, my bestie: REEEEEEEEEE.

To Carson, Harry, Will, Eric, and Marcus, for steadying me when I doubted everything, when I lost all motivation, when I didn't know what to do next, I thank you all.

Finally, to Mom, Dad, Grandma, Jess, and Sean, you are my bedrock. I would be nothing without you and your unconditional support. I love you all more than words can describe.

Licence

Contents

1 Introduction **14**
- 1.1 Background . . . 14
- 1.2 Research Questions . . . 16
- 1.3 Aim . . . 17
- 1.4 Thesis structure . . . 18
- 1.5 Contributions . . . 19
- 1.6 Associated publications . . . 20
  - 1.6.1 Journal Articles . . . 20
  - 1.6.2 Conference Papers . . . 20

2 Music Production and Audio Effects **21**
- 2.1 Music Production and Multitrack Mixing . . . 21
- 2.2 Levels and Panning . . . 23
- 2.3 Equalisation . . . 24
  - 2.3.1 FIR EQs . . . 25
  - 2.3.2 IIR EQs . . . 26
  - 2.3.3 Parametric EQs . . . 26
  - 2.3.4 Graphic EQs . . . 27
- 2.4 Reverberation . . . 27
  - 2.4.1 Convolutional Algorithms . . . 29
- 2.5 Distortion . . . 29
  - 2.5.1 Harmonic Distortion and Inter-modulation . . . 29
  - 2.5.2 Memoryless Distortion . . . 30
- 2.6 Dynamic Range Compression . . . 30

3 Machine Learning for Audio **33**
- 3.1 Machine Learning . . . 33
- 3.2 Music Information Retrieval . . . 33
  - 3.2.1 Short-Time Fourier Transform . . . 34
- 3.3 Artificial Neural Networks . . . 35
  - 3.3.1 Stochastic Gradient Descent and Backpropagation . . . 37
  - 3.3.2 Cost Functions . . . 37
  - 3.3.3 Activation Functions . . . 39
  - 3.3.4 Convolutional Neural Networks . . . 40

3.4 Perceptual Evaluation of Audio Algorithms 41
3.5 Audio Effects Modelling 42
3.5.1 Whitebox Modelling 42
3.5.2 Blackbox Modelling 43
3.5.3 Greybox Modelling 44
3.5.4 Differentiable Digital Signal Processing 44
3.6 Automatic Multitrack Mixing with Neural Networks 46
3.6.1 Intelligent Drum Mixing with the Wave-U-Net 46
3.6.2 Differentiable mixing console of neural audio effects 47
3.6.3 Automatic music mixing with deep learning and out-of-domain data . . 47
3.7 Reverse Engineering Mixes 48
3.7.1 Reverse Engineering DJ Mixes 48
3.7.2 Reverse Engineering Two Stage Cascade Mixes 49
3.7.3 The Reverse Engineer of a Mix 50

4 Time-Invariant FIR Audio Effect Matching **51**
4.1 Gain and Pan Matching 51
4.2 EQ Matching 52
4.2.1 FIR EQ Matching 52
4.3 Reverb Matching 53
4.3.1 Case 1: No truncation 53
4.3.2 Case 2: Truncation 54

5 Time-Invariant IIR Audio Effect Matching **57**
5.1 Random Polynomials 58
5.2 IIRNet Architecture Design 60
5.3 Baselines 61
5.4 Experiments 61
5.5 Evaluation Datasets 62
5.6 Results and Discussion 63

6 Memoryless Distortion Effect Matching **65**
6.1 Wiener-Hammerstein Models 66
6.2 Evaluation Dataset 66
6.3 Methods 67
6.3.1 Learnable Graphic EQ 67
6.3.2 Waveshaping nonlinearity functions 68
6.4 Optimization 70
6.5 Results 70
6.5.1 Objective Evaluation 70
6.5.2 Perceptual Evaluation 72
6.6 Discussion 73

7 Dynamic Range Compression Effect Matching **75**
7.1 Proposed DDSP DRC . . . 76
7.2 Experiment . . . 77
7.3 Initialization and Optimization . . . 78
7.4 Results . . . 79
7.5 Discussion . . . 79

8 Reverse Engineering a Linear Time-Invariant Multitrack Mix **82**
8.1 Formal Problem Statement . . . 83
8.2 Architecture and Optimization . . . 83
8.2.1 Overview . . . 83
8.2.2 EQ . . . 83
8.2.3 Gain and Pan . . . 84
8.2.4 Reverb . . . 85
8.2.5 Loss and Optimization Procedure . . . 86
8.3 Results and Analysis . . . 86
8.3.1 Objective Evaluation . . . 87
8.3.2 Perceptual Evaluation . . . 89
8.4 Discussion . . . 91

9 Reverse Engineering a Nonlinear Multitrack Mix **100**
9.1 Formal Problem Statement . . . 101
9.2 Architecture and Optimization . . . 101
9.3 Data . . . 102
9.4 Results . . . 103
9.4.1 Objective Results . . . 103
9.4.2 Perceptual Results . . . 103
9.5 Discussion . . . 105

10 Conclusions and further work **108**
10.1 Summary of contributions . . . 108
10.2 Further work . . . 112

List of Figures

2.1 Magnitude responses of a graphic EQ's second order sections and total magnitude response . . . 28
2.2 Characteristic curve of a DRC . . . 32
2.3 Dry and wet waveform of a test signal compressed with attack and release . . . 32

3.1 Hann window and its frequency response . . . 35
3.2 Spectrogram of a speaker saying "Free as Air and Water" . . . 36

4.1 Various reverb IR estimates in Case 1 . . . 54
4.2 Various reverb IR estimates in Case 2 . . . 56

5.1 Block diagram of IIRNet. . . . 58
5.2 Root placements of 100 randomly sampled 32^{nd} degree polynomials. . . . 59

6.1 Block diagram of the proposed DDSP W-H distortion effect. . . . 66
6.2 Example of estimated waveforms for each waveshaping model on a VTB example 71
6.3 Box and whisker plots of the median, standard deviation and 95% confidence intervals of all participant ratings for each waveshaping method across all plugins. No waveshaping method was found to be perceptually indistinguishable from the reference. . . . 73
6.4 Example of learned parameters using the SumTanh waveshaping model on the VTB stimulus. (A) Pre-emphasis filter (top) and de-emphasis filter (bottom), (B) Learned waveshaping function . . . 74

7.1 Block diagram of proposed DDSP DRC . . . 76
7.2 Waveforms for the dry, compressed, and reverse engineered test signals proposed by Bitzer et al. [2006] . . . 79
7.3 Loudness curves for the compressed test signal, the reverse engineered signal, and the residual between the two. . . . 80
7.4 Waveforms for the dry, compressed, and reverse engineered speech signal. . . . 80
7.5 Loudness curves for the compressed speech signal, the reverse engineered signal, and the residual between the two. . . . 81

8.1 Mixing chain diagram for the "stereo bus" architecture. In this architecture, only two reverb IRs are reverse engineered, regardless of how many stems are present in the multitrack. . . . 94

8.2 Mixing chain diagram for the "individual bus" architecture. In this architecture, each stem learns its own left and right channel reverb IRs. Thus in this diagram, a total of $2K$ reverb IRs are reverse engineered. 95

8.3 Box and whisker plots of the median, standard deviation and 95% confidence intervals of listener ratings broken down by reverse engineering architecture. The mixdown reverse engineered using the stereo reverb bus architecture was found to be perceptually indistinguishable from the reference. 96

8.4 Box and whisker plots of the median, standard deviation and 95% confidence intervals of listener ratings broken down by mixdown. A full breakdown of results is in Sec 8.3.2 96

8.5 Box and whisker plots of the median, standard deviation and 95% confidence intervals of listeners with audio production experience ratings broken down by architecture. Within this subset of listeners it was again found that the mixdowns reverse engineered using the stereo bus reverb architecture are perceptually indistinguishable from a reference. 97

8.6 Box and whisker plots of the median, standard deviation and 95% confidence intervals of listeners with audio production experience ratings broken down by mixdown. A full breakdown of results is in Sec 8.3.2 97

8.7 Learned EQs and reverbs for "Desert" using the individual bus model. The individual bus model learns an individual reverb IR for each stem, thus six reverb IRs are presented for three stems. In this case a very dense reverb IR with little decay was learned for Synth6, a separate dense reverb with delay for Synth3, and practically no reverb for Vox2. 98

8.8 Learned EQs and reverbs for "Desert" using the stereo bus model. The stereo bus model learns a left and right reverb IR for the whole mixdown, thus two reverb IRs are presented for three stems. A reverb IR with reasonable decay and echo was learned for the mixdown. 99

9.1 Signal chain of full reverse engineering system. 101

9.2 Box and whisker plots of the median, standard deviation and 95% confidence intervals of all participant ratings across all mixes. The listening test found that the mixdowns reverse engineered using the whole mixing chain were indistinguishable from a reference. 104

9.3 Learned EQ of kick drum raw track in mix 20_C. The slight boost in frequencies between 60Hz and 300Hz is associated with "boominess." 107

9.4 Log frequency spectrogram of the raw kick track and processed kick stem for mix 20_C. This demonstrates the reverse engineered effects removed most content above 250Hz and increased spectral density below 250Hz. 107

List of Tables

3.1 MSS loss measured between various signals and their copy -3dB using different sets of FFTs. 38

5.1 Average dB MSE for IIRNet trained using different families of random 16$^{\text{th}}$ order ($K = 8$) filters. 62
5.2 Comparison of average dB MSE and runtime in milliseconds fitting 16$^{\text{th}}$ order filters with other IIR filter design methods. 63
5.3 dB MSE evaluating IIRNet with different estimation orders. 63

6.1 Plugins used in this work 67
6.2 Plugin settings used to generate the dataset 67
6.3 Mean multiscale spectrogram loss of the waveshapers evaluated across pedals . 71
6.4 Results of pairwise comparison of waveshaping method architecture on perceptual similarity rating across all stimuli, with Bonferroni Correction, o $>$ 0.9, *<0.001 ·=no comparison. No waveshaping method was found to be perceptually indistinguishable from the reference. 72

7.1 DRC parameters used in ReaComp VST and learned from gradient descent on ballistics profiling signal proposed in Bitzer et al. [2006]. 78
7.2 DRC parameters used in ReaComp VST and learned from gradient descent on speech signal. 78

8.1 Songs chosen for mixing and shortened names used in this paper. 87
8.2 Average relative errors by feature subgroup for approximated mixdowns compared to the reference mixdown. 87
8.3 F stat and p-values for each multitrack and all multitracks, including all participants. 88
8.4 F stat and p-values for each multitrack and all multitracks, including only participants with audio production or mixing experience. 88
8.5 Results of pairwise comparison of mixdown architecture on perceptual similarity rating across multitracks, with Bonferroni Correction, o $>$ 0.05, *<0.001 ·=no comparison 89
8.6 Results of pairwise comparison of mixdown architecture on perceptual similarity rating within the "Etc" multitrack, with Bonferroni Correction, o $>$ 0.05, *<0.001 ·=no comparison 89

8.7 Results of pairwise comparison of mixdown architecture on perceptual similarity rating within the “Pony” multitrack, with Bonferroni Correction, o $>$ 0.05, *$<$0.001 ·=no comparison 90

8.8 Results of pairwise comparison of mixdown architecture on perceptual similarity rating within the “Desert” multitrack, including only participants with audio production experience with Bonferroni Correction, o $>$ 0.05, *$<$0.001 ·=no comparison 90

9.1 Multiscale spectrogram losses for reverse engineered mixes. Rows in bold denote the top two performing reverse engineered mixes using the full system for each song. 103

9.2 Results of pairwise comparison of mixdown architecture on perceptual similarity rating across multitracks, with Bonferroni Correction, o $>$ 0.05, *$<$0.001 ·=no comparison. The listening test found that the mixdowns reverse engineered using the whole mixing chain were indistinguishable from a reference. 105

List of abbreviations

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DAW	Digital Audio Workstation
DDSP	Differentiable Digital Signal Processing
DFT	Discrete Fourier Transform
DRC	Dynamic Range Compressor
EQ	Equaliser
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
HRTF	Head-Related Transfer Function
IDFT	Inverse Discrete-Time Fourier Transform
IIR	Infinite Impulse Response
IMP	Intelligent Music Production
IR	Impulse Response
ISTFT	Inverse Short-Time Fourier Transform
LReLU	Leaky Rectified Linear Unit
LSE	Least Square Estimate
LTI	Linear Time-Invariant
MAE	Mean Absolute Error
MIR	Music Information Retrieval
MLP	Multilayer Perceptron
MSE	Mean Square Error
MSS	Multi-Scale Spectrogram
MYW	Modified Yule-Walker
PMF	Probability Mass Function
ReLU	Rectified Linear Unit
RIR	Room Impulse Response
RMS	Root-Mean-Square
SGD	Stochastic Gradient Descent
STFT	Short-Time Fourier Transform
W-H	Wiener-Hammerstein

Chapter 1

Introduction

1.1 Background

The field of music production is incredibly broad, with research journals, conferences, and university programs dedicated to its study and advancement. To quote Burgess [2013]

> Music production is the technological extension of composition and orchestration. It captures the fullness of a composition, its orchestration, and the performative intention of the composer(s). In its precision and inherent ability to capture cultural, individual, environmental, timbral, and interpretive subtleties along with those of intonation, timing, intention, and meaning (except where amorphousness is specified), it is superior to written music and oral traditions. Music production is not only representational but also an art in itself.

The new millennium has seen an explosion of computational approaches to the study of music production, due in part to the decreasing cost of computation and the increase of digital music production techniques. The rise of digital recording equipment, MIDI, digital audio workstations (DAWs), and software plugins for audio effects led to the digital capture of various processes in music production [Burgess, 2014]. The discretization and transfer of music production into the digital realm invites its measure along some numerical dimension, and this measurement is a form of production in and of itself [Foucoult, 1975].

During this period the fields of music information retrieval (MIR) and machine learning (ML) have seen significant advancements that have provided methods for measuring and emulating portions of music production. Early machine learning experiments demonstrated that artificial neural networks (ANNs) could be used for music composition [Mozer, 1994, Eck and Schmidhuber, 2002], with more recent advancements acting as a co-composer and extending compositions [Dinculescu et al.]. Digital audio synthesis has expanded from simple additive methods such as those presented in Risberg and Shapiro [1981] to the synthesis of minute-long mp3 files [Broek, 2021]. For tasks related to multitrack mixing, a quick survey of the market sees both MIR and ML employed in software plugins such as iZotope's production suite, hardware such as Digico's mixing consoles, and music production platforms such as LANDR.

These latter algorithms can be roughly lumped into a body of research referred to as "Intelligent Music Production," (IMP) which is interested in automating portions of the music production process. Some mix engineers maintain that portions of the mixing process are ripe for automation because they are rote, time-consuming, and require little creative thought. To quote White [2008], “there’s no reason why a band recording using reasonably conventional instrumentation shouldn’t be EQ’d and balanced automatically by advanced DAW software.” According to the definition provided by De Man et al. [2019], “intelligence” refers to the automatic parameter configuration based on the analysis of the signals these systems work on (often driven by machine learning), and “music production” refers roughly to multitrack mixing as defined by effects processing applied to a multitrack recording to produce a mixdown. In the introduction of De Man et al. [2019], the authors state that “novel, multi-input multi-output (MIMO) audio signal processing methods are required, which can analyze the content of all sources to then improve the quality of capturing, altering and combining multitrack audio.”

Recently published literature has demonstrated that ANNs constitute a family of these novel MIMO methods. Examples such as [Stein et al., 2010, Martínez Ramírez et al., 2021, Martínez-Ramírez et al., 2022] present algorithms that transform a multitrack to a mixdown with no input from a mix engineer. These neural networks extract information about individual tracks of the multitrack, combine this information, and use it to apply audio effects across the multitrack and produce a mixdown. If a machine can successfully imitate the mix engineering process, what further lines of inquiry exist for researchers looking to push the field of IMP?

Two research opportunities immediately present themselves when diving into how these systems work. First, the audio processing applied to the multitrack by these algorithms do not allow for the intervention of a mix engineer. Each of these algorithms performs effects processing using blackbox audio effects, which means they use neural network-based computations to produce a mixdown rather than traditional digital signal processing algorithms. Case in point, the neural network in [Martínez Ramírez et al., 2021] mixes a multitrack using cropping, concatenation, and convolutional operations. Though the mixdowns produced by the neural network are indistinguishable from mixdowns that use traditional effects like dynamic range compression and equalisation, a mix engineer would have a hard time making adjustments to the machine’s mix due to its elaborate computations.

Second, these models have not been used to explain tendencies or trends in multitrack mixing. Upon inspection, these ANNs contain an implicit model of mixing behaviour that is the result of being trained on some dataset – given a multitrack, the ANN will produce a mixdown by applying effects based on some knowledge it has distilled from the dataset it was trained on. If machine learning is powerful enough to imitate multitrack mixing in some broad sense, can it be used to numerically characterize mixing behaviours and tendencies?

MIR has found applications towards answering this second question. In Wilson and Fazenda [2016], the authors perform MIR feature extraction on a dataset of mixdowns and analyze how these features vary across mixes. While there is some disagreement in the literature regarding how this approach may generalize [Colonel and Reiss, 2019], the core idea remains illuminating – if a model of mix engineering behaviour exists in datasets of multitracks and their mixdowns, how best can it be extracted?

An ethnographic approach to this question was presented in [Pras et al., 2018]. Here, the authors gathered data from mixing sessions made by five students in the Paris Conservatoire’s mix engineering program. The authors noted the order of tasks the mix engineers performed, how long these tasks took, the usage of audio effects, and the manner in which this processing was applied (i.e. analog effects, DAW, etc). Low-level information regarding the relative loudness of sonic elements in the mix was gathered as well, made possible by the researchers collecting the students' DAW sessions and printing the individual stems. Though the researchers presented insightful characterization of these students' mixing behaviours, there exists a roadblock when looking to scale up this approach to numerically model mixing. The collection and maintenance of DAW sessions is difficult and labor intensive, as version updates, software licenses, and quirks across platforms can all prevent access to the mixing sessions. The introduction of any analogue equipment or non-stock software plugins into the mixing chain would only compound these issues.

Another approach, called "reverse engineering a mix," was proposed in Barchiesi and Reiss [2010]. Given a multitrack and a mixdown this technique can match the effects used to mix the multitrack, with certain limitations. For example, this method does not attempt to match any reverb that may be used in a mixdown. Moreover, the paper presents separate methods for matching linear processing and nonlinear processing with no explicit method for combining the two into a full mixing chain. However, this approach does sidestep the need for DAW sessions as it makes no assumptions about how the mix was produced, e.g. if analogue technology was used, or what DAW the session may have been mixed in.

Given the recent developments in machine learning for audio mentioned above, it is worthwhile to revisit reverse engineering a mix with modern techniques.

1.2 Research Questions

The overarching question that underpins this thesis is

> **How can recent advances in machine learning for audio improve the technique of reverse engineering a mix?**

Barchiesi and Reiss [2010] mentions that one way to improve the reverse engineering algorithm would be to jointly solve for the linear and nonlinear processing in the mixdown. The software provided by the authors can reverse engineer either the dynamics processing of a mixdown, constrained to dynamic range compression, or the linear processing of a mixdown, constrained to a finite impulse response equaliser. **Can a new approach to reverse engineering a mix simultaneously optimize the linear and nonlinear processing of the mix?**

When evaluating improvements to this approach, the ultimate goal is to reverse engineer a mix within perceptual tolerance. Though this was not a goal explicitly stated in Barchiesi and Reiss [2010], reaching perceptual tolerance would demonstrate that the improved algorithm is sufficiently powerful to model mix engineering behaviour.

It would also be ideal for this new approach to reach perceptual tolerance with limited information about what processing was used to produce the target mixdown. A successful improvement to the reverse engineering algorithm would be able to match a mix produced by a human mixer using methods of their own choosing within perceptual tolerance, regardless of the software or hardware employed. Thus, this work asks **can a mix be reverse engineered in the absence of information about how the mixdown was produced?** A successful answer to this question may help to expand and scale research like Pras et al. [2018] by avoiding the need to collect and maintain DAW sessions, equipment lists, and effect parameter notes.

This leads to another question: **which audio effects need to be emulated in order to reverse engineer a mix within perceptual tolerance?** There is much research devoted to the emulation of individual audio effects using machine learning and neural networks. Therefore, decisions will ultimately be made regarding the composition of the mixing chain the algorithm reverse engineers.

Finally, **Can the parameters of a reverse engineered mixdown be made legible to a mix engineer?** Avoiding blackbox mixing, such as that mentioned above Martínez Ramírez et al. [2021], would be a welcome addition to the field of IMP. Designing an algorithm which applies legible audio processing in a familiar mixing chain allows for the interrogation of results by mixing practitioners, and the rendering of these sessions into standardized numerical parameters allows for their manipulation by MIR and ML techniques.

1.3 Aim

This work aims to improve and expand upon methods for "reverse engineering a mix" using recent developments in machine learning for audio. More specifically, the field of differentiable digital signal processing (DDSP) will serve as the basis for the development of the algorithms presented here. The choice of DDSP avoids the blackbox approaches to neural network multitrack mixing mentioned above.

Individual audio effects commonly employed by mix engineers will be explicitly modelled using DDSP and benchmarked. These effects will include gain, panning, equalisation, memoryless distortion, dynamic range compression, and artificial reverberation. These effects will be implemented in a greybox manner, which means that their parameters will be interpretable to a mix engineer. As standalone algorithms, these modules contribute to a large body of DDSP literature and may have broad application to tasks outside of mix modelling.

Ultimately these effect modules will be combined to recreate a mixing chain and used to reverse engineer mixes. Various configurations of these mixing chains will be tested against human made mixdowns with minimal restrictions to test their modelling capabilities. Listening tests will be conducted on the outputs of these systems to determine whether or not they match targets within perceptual tolerance.

1.4 Thesis structure

Chapter 2 provides necessary background of music production for the dissertation. This includes descriptions of music production, multitrack mix engineering, and audio effects.

Chapter 3 outlines machine learning for audio. This includes definitions of machine learning, gradient descent, neural networks, and music information retrieval. A literature review is also presented on differentiable digital signal processing, audio effects modelling with machine learning, and automatic multitrack mixing with neural networks.

Chapter 4 presents methodologies for matching time-invariant finite impulse response audio effects. A literature review is presented on how to match gain, panning, and equalisation. Traditional methods for artificial reverberation matching are compared against newer DDSP techniques.

Chapter 5 presents a neural network for matching time-invariant infinite impulse response audio effects. Families of random polynomials used to train this neural network are detailed. The performance of the neural network is compared to traditional algorithms for matching infinite impulse response filters.

Chapter 6 presents a novel DDSP waveshaping distortion effect. A brief definition of Wiener-Hammerstein models is provided to explain the design of the distortion effect. Several families of differentiable waveshaping functions are proposed. The difference in modelling performance of these families is measured on a reverse engineering of distortion task. Notes on the initialization of the effect for this task are provided. Both objective measures and perceptual measures are provided.

Chapter 7 presents a novel DDSP dynamic range compression effect. The notion of an approximate moving average filter is presented and discussed. The objective performance of the effect is measured on a reverse engineering of dynamic range compression task. Notes on the initialization of the effect for this task are provided.

Chapter 8 defines the task of reverse engineering a multitrack mix. A method for reverse engineering a mix using only linear time-invariant effects is presented, which employs a DDSP mixing chain comprised of gain, panning, equalisation, and reverb. Notes on the initializations for each module in the mixing chain are provided. The performance of different mixing chains are compared to one another and measured using both objective and perceptual tests.

Chapter 9 presents a methodology for reverse engineering a mix using both linear and non-linear effects. A DDSP mixing chain comprised of gain, panning, equalisation, reverb, waveshaping distortion, and dynamic range compression is shown. Notes on the initializations for each module in the mixing chain are provided. The performance of different mixing chains are compared to one another and measured using both objective and perceptual tests.

Chapter 10 concludes the dissertation and presents directions for further work.

1.5 Contributions

Contributions of this thesis are:

- Chapter 5: A neural network architecture IIRNet is presented which can match an arbitrary magnitude response to a cascade of biquads.
- Chapter 6: A differentiable Wiener-Hammerstein model is presented that can match a memoryless distortion effect within perceptual tolerance.
- Chapter 7: A differentiable dynamic range compressor with ballistics is presented that uses approximate moving average filters to match attack and release times.
- Chapter 8: A differentiable linear time-invariant mixing chain is demonstrated to match LTI multitrack mixes within perceptual tolerance.
- Chapter 9: A differentiable nonlinear time-invariant mixing chain is proposed that can match nonlinear time-invariant mixes within perceptual tolerance, and an ablation study demonstrates how objective accuracy is best with a mixing chain comprised of gain, panning, equalisation, dynamic range compression, distortion, and reverb.

1.6 Associated publications

Portions of the work detailed in this thesis have been presented in national and international scholarly publications, as follows:

1.6.1 Journal Articles

- Colonel, Joseph T., and Joshua Reiss. "Reverse engineering of a recording mix with differentiable digital signal processing." The Journal of the Acoustical Society of America 150.1 (2021): 608-619.

  Contains portions of Chapter 8

1.6.2 Conference Papers

Peer Reviewed

- Colonel, Joseph, et al. "Reverse engineering memoryless distortion effects with differentiable waveshapers." Audio Engineering Society Convention 153. Audio Engineering Society, 2022.

  Contains portions of Section 2.5 & Chapter 6

- Colonel, Joseph T., et al. "Direct design of biquad filter cascades with deep learning by sampling random polynomials." ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2022.

  Contains portions of Section 2.3 & Chapter 5

Extended Abstract Peer Reviewed

- Colonel, Joseph, and Reiss, Joshua. "Approximating ballistics in a differentiable dynamic range compressor." Audio Engineering Society Convention 153. Audio Engineering Society, 2022.

  Contains portions of Section 2.6 & Chapter 7

- Colonel, Joseph, and Joshua D. Reiss. "Exploring preference for multitrack mixes using statistical analysis of mir and textual features." Audio Engineering Society Convention 147. Audio Engineering Society, 2019.

  Contains portions of Section 3.2

Chapter 2

Music Production and Audio Effects

This chapter describes the process of music production with a focus on mix engineering. With this background established, common audio effects used in mix engineering are listed and their formulations and usages are detailed.

2.1 Music Production and Multitrack Mixing

The notion of a "mix engineer" developed from the introduction of magnetic tape recording technology to recording studios. From Hepworth-Sawyer and Hodgson [2016]:

> The creation of magnetic tape and its adoption within the studio environment meant that no longer were engineers capturing the performance that would effectively serve as the master for production (production here meaning manufacture of the listener's product). As soon as magnetic tape made an entrance, it was necessary for a new stage, a new task, whether undertaken by the same person as the recording, whereby the audio material needed to be transferred or 'cut' to vinyl.

From its genesis, mix engineering has concerned itself with the transformation of a multitrack recording into a mixdown, often a stereo or mono recorded medium. In the modern context, the Berklee College of Music's career website presents the following scenario to describe the work of a mix engineer [1]:

> After receiving tracks from the recording engineer following a recording session, the mixing engineer focuses on three main tasks: manipulating the volume levels to emphasize the important elements in each section, enhancing the sonic character of each track with the aid of processers such as EQ and compression, and adding effects like delay and reverb. [...] When all parties are satisfied, the final mix of the recording—or "mixdown"—is printed and delivered to the client.

[1] https://www.berklee.edu/careers/roles/mixing-engineer, Accessed 20 April, 2023

Taken for granted in this description is the typical division of labour seen in major recording studios of the global North. Within this framework a recording engineer would be responsible for the setup of microphones, a mastering engineer responsible for mastering the mixdown, and a music producer overseeing and directing the project. Yet these neat divisions are rarely encountered outside of such studios.

Over the course of the 20th and 21st centuries, technological advances have blurred the lines between traditional roles in music production. The advent of cheap recording equipment, powerful computer hardware, digital audio workstations (DAW), widely available plugins, and decentralised knowledge sharing has created a scenario where someone can be composer, performer, recording engineer, mix engineer, and mastering engineer from the comfort of their bedroom. More astonishingly, recent advancements in machine learning and artificial intelligence can automate this entire process [Najduchowski et al., 2018, Civit et al., 2022].

Nevertheless, many researchers continue to define mix engineering as the application of audio effects to a multitrack in order to produce a mixdown. What is lost, and what is gained, by using this simplification?

As Pras et al. [2019] observes, this definition of "mixing" is far from universal. Case in point, DAW practitioners of music studios in Bamako, Mali frequently rearrange their compositions, swap MIDI presets, and mute certain repetitions throughout a recording session. However, this process is referred to as "mixing" once vocals have been recorded. Other techniques left out of a simplified definition of mixing include take selection, time alignment of sources, pitch correction, and the various soft skills employed during a recording session.

However, this shared understanding of mixing held by mix engineers of the global North is worthy of further inquiry; this flattening of mix engineering to effects processing has led to a rich body of research. Several dissertations' worth of research has been published that seek to interrogate, categorize, and model this restricted definition of mix engineering. For example, one PhD dissertation presented interrogations of mixing "best practices and common sense [Pestana, 2013]." This work presented 88 assumptions regarding mixing, and interviewed 49 mix engineers to ascertain the validity of these assumptions. Moreover, five mix engineers were asked to mix excerpts of two separate multitracks using a specific methodology: beginning with level setting, followed by panning, then EQ, then compression, then reverb, and finally automation.

Another such thesis, "Towards a Better Understanding of Mix Engineering [De Man, 2017]," provides much of the framework for this dissertation. In it, the author produces a "Mix Evaluation Dataset" in order to explore differences in the effects processing of different mix engineers. Mix engineers contributed to this dataset by providing mixdowns of pre-selected multitracks as well as commenting on their peers' mixdowns in a blind fashion. This restriction of "mix engineering" allows for computationally driven inquiries of the semantic descriptors found in the dataset [Moffat et al., 2022]. Moreover, this framework has allowed for the measurement of cultural differences in mix engineering and listening habits between Western and Japanese mix engineers [Tajima and Kawahara, 2019].

As such, this dissertation will adopt a similar definition of mix engineering as that explored in De Man [2017]. This definition, adapted from Izhaki [2008], defines mix engineering as "a complex task that includes dynamically adjusting levels, stereo positions, filter coefficients, dynamic range processing parameters, and effect settings of multiple audio streams." **Thus,**

when looking to model music production behaviour, we are specifically seeking to model the effects processing used by mix engineers. We posit that though this is a rather strict definition, it allows for a broad treatment and deep line of inquiry.

According to Izhaki [2008] a mix engineer can transform a multitrack in this fashion by targeting dimensions such as the mix's

- Mood – the emotional context of the music in the mix
- Balance – the perceived affect of frequencies, stereo image, and levels in the mix
- Definition – how distinct and recognizable sounds are within the mix
- Interest – accommodations to the listener's fluctuations in attention across the mix

2.2 Levels and Panning

Levels are often the first consideration when mixing a multitrack. It is the job of the mix engineer to ensure each sonic element of the multitrack is presented in a mix according to its importance, and a strong level balance is the first step in determining which elements of the multitrack should exist in the foreground [Case, 2011].

In order to achieve a rough perceptual balance, the mix engineer will apply multiplicative gains to the individual tracks in the multitrack. Various scales and regimes for gain staging may be used depending on the interface used by the mix engineer. Traditional mixing consoles provide potentiometer-based or vca-based faders, which provide tactile feedback when gain staging (move the fader up, the gain goes up) [Izhaki, 2008]. Typically, these faders display gain on the decibel (dB) scale, with 0dB applying unity gain and $-\infty$dB applying 0 gain (or muting the track). A dB scaled fader cannot apply a negative gain to track. In order to apply a negative gain, polarity inversion can be used in conjunction with the dB fader.

Panning refers to moving individual elements in the stereo field of the mix, or where these individual elements will be perceived on a horizontal plane in front of the listener. It is possible to pan tracks by setting faders that correspond to the level sent to the left and right channel of a stereo mix. However, it is far more common to employ pan pots for panning [Izhaki, 2008]. When a pan pot is set to its farthest position counterclockwise (-1) the track is panned entirely to the left (i.e. a gain of 0 is applied before sending the track to the right channel), and when a pan pot is set to its farthest position clockwise (+1) the track is panned entirely to the right. What happens between these positions varies across pan pot designs.

Both physical and digital pan pots use various pan laws to determine the left and right channel gains for a track. The 0dB pan law does not drop the level of centrally panned signals, i.e. both the left and right channel receive the track with unity gain. This can lead to a perceived 3dB boost in the perceived signal loudness compared to a signal panned hard left or hard right. Much more commonly used is the -3dB pan law [De Man and Reiss, 2013], in which centrally panned signals are dropped 3dB to compensate for the perceived boost of the 0dB pan law. The -3dB pan law, also referred to as the equal power or sine/cosine law, smoothly increases the gain applied to the signal so that equal loudness is achieved across the stereo field. With panning value $p \in [-1, 1]$, the left gain g_L and right gain g_R can be calculated with

$$\begin{aligned} g_L &= cos(\frac{\pi(p+1)}{4}) \\ g_R &= sin(\frac{\pi(p+1)}{4}) \end{aligned} \tag{2.1}$$

2.3 Equalisation

Equalisation (EQing) shapes the frequencies of a track and can be used across a multitrack to achieve a spectral balance in a mix. EQing is also often used to help separate the elements in a mix, as the frequency ranges of varying instruments can overlap one another. By cutting or boosting certain frequencies on given tracks the mix engineer can avoid masking, which Miller [1947] defines as "the shift of the threshold of audibility of the masked sound due to the presence of the masking sound." In other words, the ability to discern one element in a mix may be clouded by the presence of another element in the mix that occupies a similar frequency range.

Digital EQs with no parameter automation can be considered linear time-invariant (LTI) systems, which means their behaviour can be fully characterized by their impulse response [Oppenheim et al., 1997]. In the digital domain, the impulse response for an LTI system can be calculated by convolving the LTI filter with the digital Dirac function

$$\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & otherwise \end{cases} \tag{2.2}$$

The z Transform

Often the frequency response of these systems is of interest. To gain an intuitive understanding of this, the z Transform and discrete Fourier transform (DFT) must be defined. The following definitions are adapted from Rabiner and Gold [1975].

Given a sequence $x(n)$ defined for all n, its z transform is defined as the complex valued function

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n} \tag{2.3}$$

where z is the complex variable. In the case where $x(n)$ is nonzero on some finite interval $[N_1, N_2]$, $X(z)$ converges everywhere in the complex plane with the possible exceptions of $z = 0$ or $z = \infty$. For causal signals, i.e. $x(n) = 0$ for $n < 0$, $X(z)$ converges everywhere outside a circle of radius R_1 in the complex plane. R_1 depends on the singularities, or poles, of $X(z)$, and if $R_1 < 1$ the system is said to be stable.

The frequency response of a filter is defined as the evaluation of its z transform along the unit circle

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \tag{2.4}$$

This continuous, complex valued function can be decomposed into its magnitude response $|X(\omega)|$ and its phase response $\angle X(\omega)$.

When a sequence $x(n)$ is convolved with a sequence $h(n)$ to produce $y(n)$, their z transforms multiply, i.e.

$$Y(z) = X(z)H(z) \tag{2.5}$$

Note here that $|Y(\omega)| = |X(\omega)||H(\omega)|$ and $\angle Y(\omega) = \angle X(\omega) + \angle H(\omega)$.

Discrete Fourier Transform

The DFT $X_p[k]$ of a periodic signal $x_p[n]$ with period N is the complex-valued function

$$X_p[k] = \sum_{n=0}^{N-1} x_p[n]e^{-j(2\pi/N)nk} \tag{2.6}$$

An important connection exists between the z transform and the DFT. The periodic sequence created by using as its DFT coefficients the values of the z transform evaluated at N points around the unit circle of a nonperiodic sequence is an aliased version of the nonperiodic sequence.

EQs can be divided into finite impulse response (FIR) designs and infinite impulse response (IIR) designs. As their names suggest, FIR EQs have an impulse response of finite duration and IIR EQs have an impulse response of infinite duration. A comprehensive literature review of equalisation effect designs and implementations is presented in Välimäki and Reiss [2016].

2.3.1 FIR EQs

A causal discrete FIR EQ of order N is formulated as

$$\begin{aligned} h[n] &= b_0\delta[n] + b_1\delta[n-1] + \cdots + b_N\delta[n-N] \\ &= \sum_{i=0}^{N} b_i\delta[n-i] \end{aligned} \tag{2.7}$$

with b_i the i^{th} value of the filter's impulse response.

FIR EQs have several desirable properties. First, these filters are stable by design, which means a finite input will guarantee a finite output. Second, these filters have no feedback which can help avoid rounding errors. Third, these filters can be easily designed to be linear phase and thus avoid phase distortions.

To design an FIR filter with a desired frequency response, one may use the frequency sampling method. As summarized by Rabiner and Gold [1975], "to approximate any continuous frequency response, one could sample in frequency at N equispaced points around the unit circle (the frequency samples) and evaluate the continuous frequency response as an interpolation of the sampled frequency response." To calculate the time-domain filter, one takes the inverse DFT of the sampled frequency response [Smith, accessed 03 November, 2022]

$$x_p[n] = \frac{1}{N}\sum_{k=0}^{N-1} X(\omega_k)e^{j\omega_k n} \tag{2.8}$$

2.3.2 IIR EQs

Infinite impulse response (IIR) filters have a variety of applications, such as control systems, time series forecasting, and audio signal processing [Dorf and Bishop, 2011, Hamilton, 1994, Välimäki and Reiss, 2016]. In audio applications, digital IIR filters are often used for equalisation, including tone matching, feedback reduction, and room compensation [Ramos and Lopez, 2006]. Classical methods for designing digital IIR filters are generally restricted to specific prototypes, e.g. designing a lowpass filter with minimum passband ripple [Selesnick and Burrus, 1998]. However, some applications require designing a filter that achieves an arbitrary magnitude and/or phase response. Classical methods for this task include the modified Yule-Walker (MYW) estimation [Chan and Langford, 1982], least squares approaches [Kobayashi and Imai, 1990, Lang, 1998], linear programming [Rabiner et al., 1974], Steiglitz-McBride [Stoica and Soderstrom, 1981], and gradient-based optimization methods [Dodds, 2020].

An N^{th} degree digital IIR filter can be characterized by its transfer function as shown in (2.9).

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_N z^{-N}}{a_0 + a_1 z^{-1} + \cdots + a_N z^{-N}} \tag{2.9}$$

For most applications, $b_i, a_i \in \mathbb{R}$. To facilitate numerical stability, these filters are often implemented as a cascade of K second order biquad section $H(z) = \prod_{k=0}^{K-1} H_k(z)$ where

$$H_k(z) = g\frac{1 + b_{1,k}z^{-1} + b_{2,k}z^{-2}}{1 + a_{1,k}z^{-1} + a_{2,k}z^{-2}}. \tag{2.10}$$

Should the poles and zeros of each biquad fall within the unit circle of the complex plane, the digital IIR filter is said to be *minimum phase*. The magnitude response of these filters $|H(e^{i\omega})|$ can be calculated by evaluating $H(z)$ along the unit circle in the complex plane, and taking the magnitude of the result.

$$|H(e^{i\omega})| = \Big|\frac{b_0 + b_1 e^{-i\omega} + \cdots + b_N e^{-iN\omega}}{a_0 + a_1 e^{-i\omega} + \cdots + a_N e^{-iN\omega}}\Big| \tag{2.11}$$

In practice, the logarithm of this magnitude response is of interest,

$$\log(|H(e^{i\omega})|) = G + \log\Big(\prod_{k=0}^{K-1} \Big|\frac{1 + b_{1,k}e^{-i\omega} + b_{2,k}e^{-2i\omega}}{1 + a_{1,k}e^{-i\omega} + a_{2,k}e^{-2i\omega}}\Big|\Big). \tag{2.12}$$

where G refers to the gain of the filter.

2.3.3 Parametric EQs

Quoting Välimäki and Reiss [2016],

> The parametric equalizer is the most powerful and flexible of the equalizer types. Midrange bands in a parametric equalizer have three adjustments: gain, center frequency, and quality factor Q (or bandwidth). A parametric equalizer allows the operator to add a peak or a notch at an arbitrary location in the audio spectrum. At other frequencies, far away from the peak or notch, the parametric equalizer does not modify the spectral content, as its magnitude response there is unity (0dB).

A parametric equaliser is made up of first- or second-order sections connected in cascade. These sections typically include second order notching/peaking filters and first- or second-order shelving filters for the lowest and highest frequencies.

Low shelving filters are designed to boost or attenuate frequencies below a specified "crossover" frequency and leave other frequencies untouched. High shelving filters do the opposite and boost or attenuate frequencies above a specified crossover frequency.

Peaking and notching filters are designed to alter frequencies within a specified band. Peaking filters boost these frequencies, and notching filters attenuate them. These filters can be designed given a center frequency, which defines the center of the frequency band the filter will alter, a quality factor Q, or the ratio of the filter's center frequency to its bandwidth, and the filter's gain G.

2.3.4 Graphic EQs

Again quoting Välimäki and Reiss [2016],

> The graphic equalizer is a tool for independently adjusting the gain of multiple frequency regions in an audio signal. [...] Structurally, a graphic EQ is a set of filters, each with a fixed center frequency and bandwidth. The only user control is the command gain, or the amount of boost or cut, in each frequency band.

Typically the bands of a graphic EQ are spaced logarithmically to mimic human perception, such as octave bands or 1/3 octave bands. A graphic EQ can be implemented either as a cascade of equalising filters or as a parallel bank of bandpass filters. Both designs suffer from interaction between neighboring filters, where a change in one command gain affects the magnitude response of a fairly wide frequency range. Figure 2.1 plots the magnitude response of a 10 octave band graphic EQ. Note that while each command gain is set to +10dB, the magnitude response of the EQ exhibits ripple and can boost certain frequencies up to +15dB.

2.4 Reverberation

Senior [2011] states that reverberation is "the most widely used sweetening effect in record production" that can provide five enhancements for a mix simultaneously:

- Blend – making disconnected tracks sound as if they belong with one another
- Size – increasing the apparent dimensions of a mix's acoustic environment
- Tone – altering the subjective tone of a track due to the phase-cancellation of echoes
- Sustain – increasing the duration of a dry sound
- Spread – spreading information across the stereo image

A comprehensive literature review of artificial reverberation effect designs and implementations is presented in Valimaki et al. [2012] and Välimäki et al. [2016]. Valimaki et al. [2012] provides a scientific definition of reverberation:

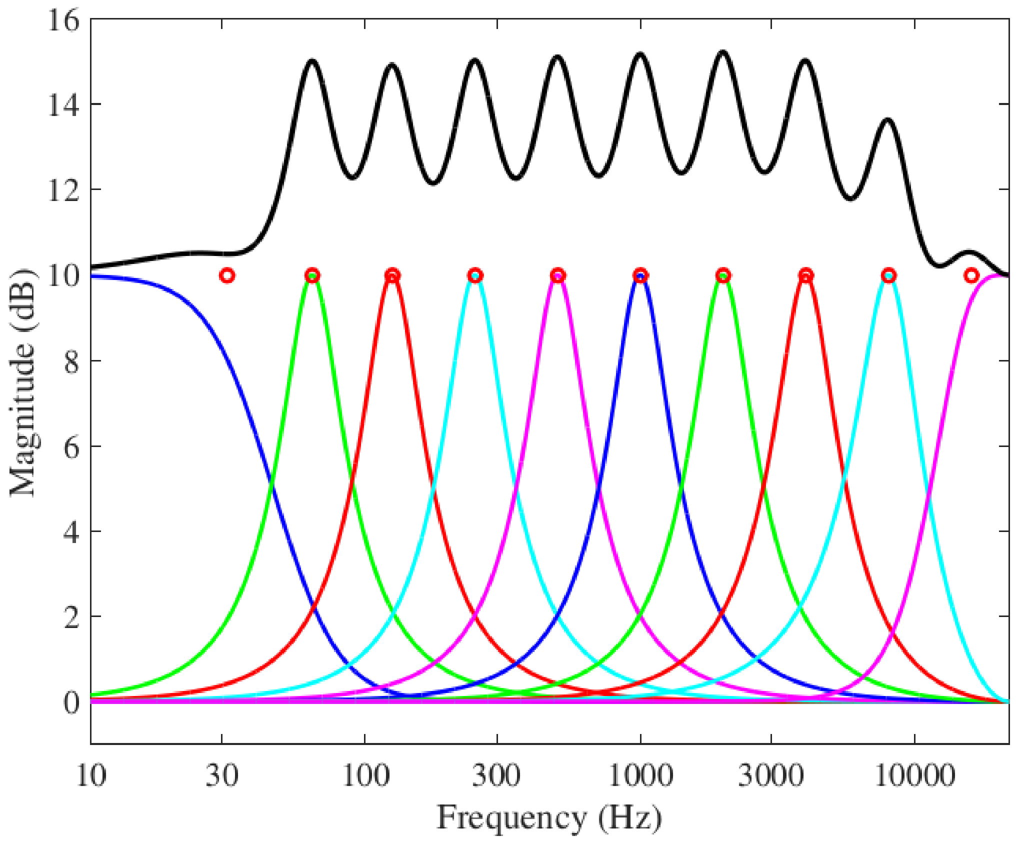


Figure 2.1: Magnitude responses of a graphic EQ's second order sections and total magnitude response

> Reverberation refers to the prolonging of sound by the environment, which is essentially caused by the reflectivity of surfaces and by the slow speed of sound in air [...]. As sound radiates from a source, it interacts with the environment, carrying with it to the listener an imprint of the space, and a sense of the objects and architecture present.

In the early days of broadcast and recording, artificial reverberation was applied to audio through various physical systems, including echo chambers, spring reverberators, and reverberation plates. It wasn't until Schroeder and Logan [1961] that the idea of artificial reverberation based on digital signal processing was proposed.

Valimaki et al. [2012] divides artificial reverberation algorithms into four classes:

- Delay networks – algorithms that simulate reverberation using delay lines, filters, and feedback connections
- Convolutional methods – algorithms that convolve a dry signal with a recorded, approximated, or simulated room impulse response (RIR)
- Computational acoustics – algorithms that simulate the propagation of sound in a specified geometry

- Virtual analog models – algorithms that simulate electromechanical or electrical devices, such as tapes, plates, or springs) used for producing reverberation effects

Each of these classes currently sees an abundance of research and investigation [Ibnyahya and Reiss, 2022, Okuzono et al., 2021, Martínez Ramírez et al., 2020]. For the purposes of this dissertation, however, only convolutional algorithms will be discussed.

2.4.1 Convolutional Algorithms

Theoretically speaking the application of a convolutional reverb is the same as an FIR EQ: an audio signal is convolved with a RIR to simulate that room's reverberation characteristics. The measurement and collection of RIRs is a field unto itself, and a thorough description falls outside the scope of this dissertation. The primary distinction between an FIR EQ and RIR is length in time – a typical FIR EQ has a length on the order of 10^4 samples, whereas a RIR may have length of 10^5 or even 10^6 samples. Case in point: a 1.0 second long RIR sampled at 48kHz requires $48,000$ multiplications and additions per output sample, or 5 billion floating-point operations per second [Valimaki et al., 2012]. Moreover, these computations quickly double as distinct IRs are typically used in the left and right channel of a stereo mix as natural reverberation becomes decorrelated by the time it reaches a listener's ear.

Advancements in processing power, especially in GPU processors, have allowed for the widespread adoption of convolutional reverb algorithms in audio production. Moreover, much research has gone into fast convolutional techniques that enable speed-ups in the application of artificial convolutional reverb. A fast Fourier transform (FFT) algorithm can reduce the the $\mathcal{O}(N^2)$ time complexity of a time-based convolution to a $\mathcal{O}(n \log n)$ multiplication in the frequency domain.

2.5 Distortion

Distortion occurs when some nonlinearity is introduced to the signal processing chain. No analogue components are perfectly linear, so technically speaking distortion is introduced whenever analogue circuitry is used in music production. More relevantly, musical distortion can be introduced by saturating the equipment used for recording or playback; because the components in audio circuitry are rated for some nominal voltage range, they can introduce obvious distortions when pushed beyond this range. According to Izhaki [2008], distortion is prominent in mixing because it can compensate for "boringly precise" digital sounds and because it can increase the aggression of a track (whether metal, pop, or dance). This type of distortion is referred to as waveshaping distortion, as it alters the waveform of a signal rather than its phase or time-domain characteristics [De Man and Reiss, 2014].

2.5.1 Harmonic Distortion and Inter-modulation

A musical tone such as that of a guitar playing a note can be characterized by its fundamental frequency and overtones. The fundamental frequency is measured as the lowest frequency of a periodic waveform, and overtones are the integer multiples of fundamental. Perceptually, the pitch of a tone is informed by the fundamental (though it is more accurate to say that pitch is informed by the lowest *perceived* harmonic, which may not be the fundamental). The overtones play a key role in the perceived timbre of a sound and, for example, can help a listener distinguish between a violin and a guitar playing the same fundamental.

As its name implies, harmonic distortion alters the harmonics of a sound [Izhaki, 2008]. Many analog distortion circuits are sought after due to the harmonic distortion their components can apply to a signal. Inter-modulation distortion effects, on the other hand, introduce additional frequencies that are not harmonically related to the input sound. When naively implemented, many digital distortion effects can introduce inter-modulation and can sound more harsh than analog effects producing harmonic distortion, though analog effect circuits can impart some amount of inter-modulation distortion as well.

Inter-modulation distortion can occur when the overtones produced by the distortion exceed the Nyquist frequency, or half the sampling rate of the audio signal, and alias. For example, if a 8kHz sinewave is sampled at 44.1kHz and distorted, its fourth harmonic at 32kHz exceeds the Nyquist frequency of 22.05kHz. This fourth harmonic will mirror around the Nyquist frequency as 12.1kHz, which is not harmonically related to the fundamental. Techniques such as oversampling and antiderivative antialiasing can be used to avoid inter-modulation [Bilbao et al., 2017].

2.5.2 Memoryless Distortion

Quoting Rouphael [2009],

> Nonlinear circuits are considered to be either memoryless or with memory. In memoryless circuits, the output of the circuit at time t depends only on the instantaneous input values at time t and not on any of the past values of its input. [...] The output of a nonlinear circuit with memory at time t, on the other hand, depends on the input signal at time t as well as inputs that occurred in previous instances before t. The largest time delay on which the output depends determines the memory of the circuit.

Trivially, a memoryless distortion can be characterized as

$$y[n] = f(x[n]) \tag{2.13}$$

where $f(\cdot)$ is some nonlinear function [Parker et al., 2016]. While much research has gone into modeling distortion effects with memory, this dissertation will only treat memoryless distortion.

2.6 Dynamic Range Compression

Dynamic range compression is an essential tool in mix engineering and audio production [Izhaki, 2008]. This nonlinear audio effect is often used to reduce the dynamic range of a signal. This is done by selectively attenuating peaks in the signal while leaving quieter portions untouched. Digital implementations of dynamic range compressors (DRC) vary, leading to much diversity in the types of controls exposed to users and the considerations taken into account when setting parameter values [Giannoulis et al., 2012].

Challenges arise in the design of digital DRC as it is a nonlinear time-dependent audio effect with memory. A typical feedforward digital DRC, as outlined in Bitzer et al. [2006], is composed of the following modules: a level detector tasked with measuring the level of the incoming signal; a gain computer tasked with calculating the instantaneous attenuation or gain to be applied to the signal; and a gain smoother tasked with modifying the output of the gain computer based on the time-varying fluctuations of the input signal's loudness. Key to the design of many DRCs are the attack and release times that control the gain smoother. These parameters help to avoid introducing distortion and control how quickly the DRC acts.

Due to the complexity of a DRC and its usage, much work has gone into characterizing, estimating, and automating its parameters. Previous literature has developed procedures and test signals for profiling DRCs [Bitzer et al., 2006], used regression models and reference signals to control a DRC [Sheng and Fazekas, 2017], developed feature-based heuristics to operate a DRC [Giannoulis et al., 2013], and implemented cross-adaptive algorithms to set DRC parameters settings across a multitrack recording [Ma et al., 2015].

The operation of a DRC can be defined using the following parameters. The analysis presented here is primarily adapted from Giannoulis et al. [2012].

- *Threshold* is the level used to determine whether or not to apply compression to the input signal. When the signal is measured above the threshold, compression is applied. In this formulation, the knee is not centered about the threshold; instead, it begins after the threshold. How the input's level is measured varies across DRC designs, with typical implementations including sample-by-sample peak detection and root-mean-square (RMS) measures.

- *Ratio* determines the amount of compression applied and is a measure of the input/output ratio for signals crossing the threshold.

- *Knee-width* is a threshold-dependent value that allows for a smooth transition in the DRC's compression characteristic curve above and below the threshold. A small value creates a sharp transition between unity gain and the compression ratio, and a larger value produces a gradual transition.

- *Makeup Gain* refers to a gain applied to the compressor's output signal.

- *Attack time* and *release time* determine how long it takes the compressor to attenuate the signal according to the ratio after surpassing the threshold and how long the compressor continues to attenuate the signal after dropping below the threshold, respectively. In many designs these parameters also control how the attenuation applied by the compressor is smoothed over time.

The characteristic curve of a DRC describes its static parameter settings and plots its input/output transfer function. An example characteristic curve is shown in Figure 2.2 with threshold set to -20dB, makeup gain $+10$dB, compression ratio $5:1$, and both hard and soft knee.

Figure 2.3 shows a test signal used to characterize the attack and release of a DRC and the result of that signal being passed through a compressor with attack time set to 40ms and release time set to 200ms. As demonstrated in the figure the DRC smoothly reaches its characteristic attenuation over several milliseconds after the test signal crosses the threshold and smoothly ceases attenuation over a longer time frame.

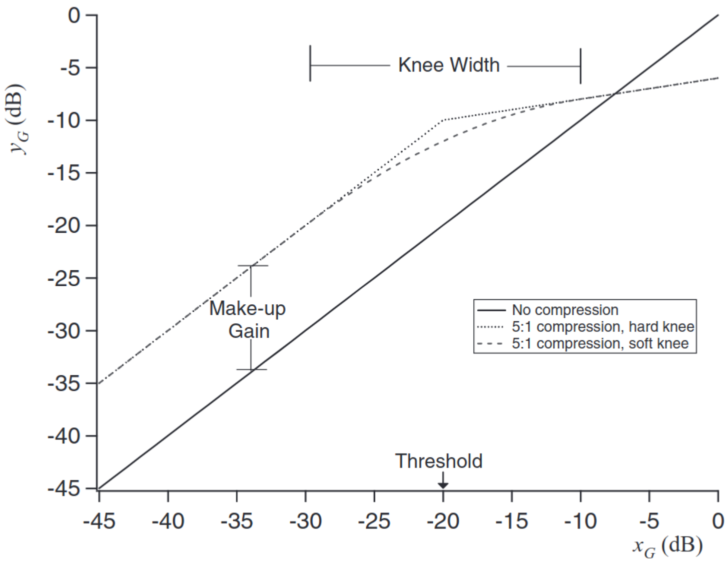


Figure 2.2: Characteristic curve of a DRC

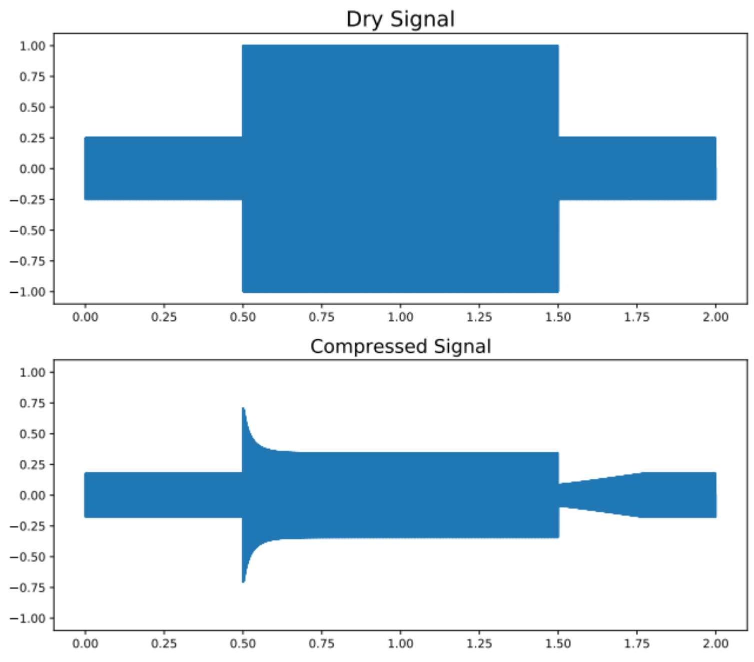


Figure 2.3: Dry and wet waveform of a test signal compressed with attack and release

Chapter 3

Machine Learning for Audio

This section defines machine learning, applications of machine learning to audio, music information retrieval, and the short-time Fourier transform.

3.1 Machine Learning

Though machine learning has come to describe a vast field of study, Mitchell and Mitchell [1997] provides a straightforward definition of what constitutes a machine learning algorithm:

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Machine learning has been applied to such tasks as learning to drive autonomous vehicles, learning to play board games such as backgammon or go, and learning to recognize spoken words.

According to Mitchell and Mitchell [1997], when designing a machine learning algorithm for some task T one must explicitly chose the training experience E, the target function which maps actions the algorithm can take to some state, a numerical representation of this target function, and a learning algorithm which can improve the target function's ability to map an experience E according to a performance measure P.

3.2 Music Information Retrieval

Many machine learning applications for audio make use of music information retrieval (MIR), though MIR has seen broad research outside of machine learning for audio. To quote Schedl et al. [2014], "MIR is foremost concerned with the extraction and inference of meaningful features from music (from the audio signal, symbolic representation or external sources such as web pages), indexing of music using these features, and the development of different search and retrieval schemes (for instance, content-based search, music recommendation systems, or user interfaces for browsing large music collections)." Downie [2003] propose the following facets that MIR attempts to categorize within music:

- Pitch facets – "the perceived quality of a sound that is chiefly a function of its fundamental frequency in—the number of oscillations per second;" the temporal facet
- Temporal facets – that which makes up the rhythmic component of a musical work
- Harmonic facets – when two or more pitches sound at the same time
- Timbral facets – all tone and color not related to pitch or harmony
- Editorial facets – performance instructions such as fingerings, ornamentation, dynamic instructions, and so on
- Textual facets – aspects such as lyrics of songs, arias, chorales, hymns, and symphonies
- Biographic facets – information concerning a work's title, composer, arranger, editor, lyric author, publisher, edition, catalogue number, publication date, discography, performer(s), and so on

Feature extraction can be used to characterize several of these facets from an audio waveform. Roughly, MIR features (such as those mentioned in Lartillot et al. [2008]) can be divided into calculations performed on an audio waveform (e.g. zero-crossing rate, RMS energy, envelope) and calculations performed on an audio signal's frequency content (e.g. spectrum, filterbank).

3.2.1 Short-Time Fourier Transform

To analyze the pitch, harmonic, and timbral facets of music, it is useful to obtain a representation of a finite discrete signal in terms of its frequency content. One useful representation is the Short-Time Fourier Transform (STFT). The STFT of a signal $x[n]$ is

$$X(n,\omega) = \sum_{m=-\infty}^{\infty} x[n+m]w[m]e^{-j\omega m} \tag{3.1}$$

where $w[n]$ is a window sequence and ω is the frequency in radians. Note that the STFT is periodic in ω with period 2π. Thus we need only consider values of ω for $-\pi \leq \omega \leq \pi$.

Often, the Hann window is chosen as $w[n]$

$$w_{Hann}[n] = \begin{cases} 0.5 - 0.5cos(2\pi n/M) & 0 \leq n \leq M \\ 0 & otherwise \end{cases} \tag{3.2}$$

Refer to Figure 3.1 for plots of the Hann window. Spectral resolution refers to the STFT's ability to distinguish two sinusoidal components of a signal with fundamental frequencies close to one another. Spectral resolution is influenced primarily by the width of the main lobe of the window's frequency response. Spectral leakage, on the other hand, refers to a window's tendency to smear a sinusoid's fundamental frequency into neighboring frequency bins. Spectral leakage is influenced primarily by the relative amplitude of the main lobe to the side lobes.

A signal's STFT can be inverted using an overlap-add procedure [Allen, 1977]. First, a windowed output frame is obtained via:

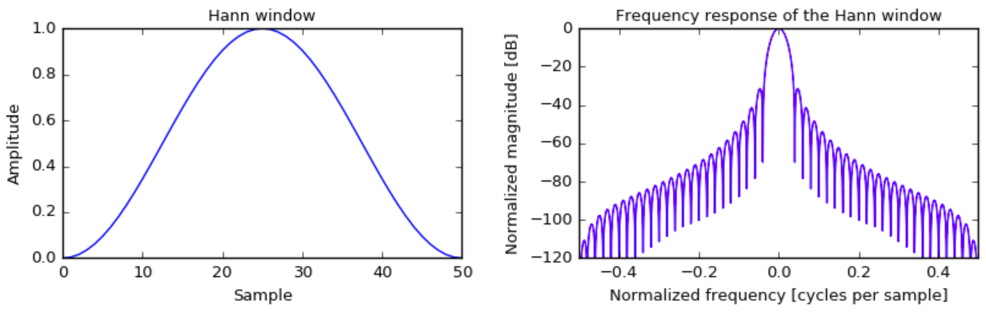


Figure 3.1: Hann window and its frequency response

$$\hat{x}'_m(n) = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \hat{X}'_m(e^{j\omega_k}) e^{j\omega_k n} \tag{3.3}$$

Then, the final output is reconstructed by overlapping and adding the windowed output frames:

$$\hat{x}(n) = \sum_m \hat{x}'_m(n - mR) \tag{3.4}$$

where R is the hop size, or how many samples are skipped between frames. This analysis and resynthesis becomes an identity operation if the analysis windows sum to unity, i.e.

$$A_w(n) \triangleq \sum_{m=-\infty}^{\infty} w(n - mR) = 1 \tag{3.5}$$

The STFT is a complex valued signal, which means that each $X(n_i, \omega_j)$ has a magnitude and phase component. We refer to a three dimensional representation of the magnitude of the STFT $||X(n, \omega)||_2$ as “the spectrogram.” The spectrogram is used throughout audio analysis because it succinctly describes a signal’s spectral power distribution, or how much energy is present in different frequency bands, over time. Refer to Figure 3.2 for an example spectrogram.

3.3 Artificial Neural Networks

Artificial neural networks (ANNs) are a class of machine learning algorithms that have found widespread usage in machine learning tasks. ANNs “provide a practical method for learning real-valued and vector-valued functions over continuous and discrete-valued attributes, in a way that is robust to noise in the training data [Mitchell and Mitchell, 1997].” The name is inspired by biological neurons and thus uses the term “neurons” to describe the individual units of the ANN that are connected to one another via some calculation.

The most straightforward ANN topology is the “feedforward neural network.” Adopting the terminology provided by Bishop [1994]:

> A feedforward neural network can be regarded as a nonlinear mathematical function which transforms a set of input variables into a set of output variables. The

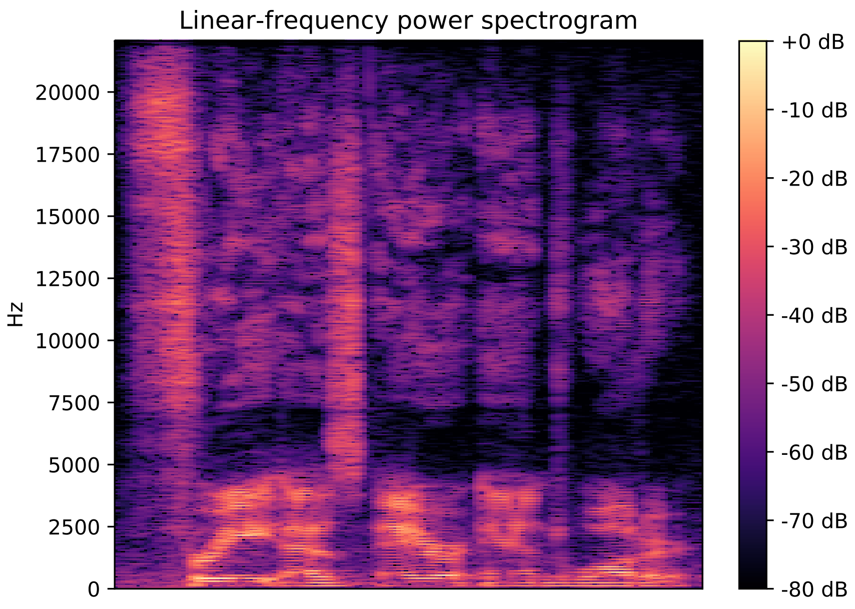


Figure 3.2: Spectrogram of a speaker saying "Free as Air and Water"

> precise form of the transformation is governed by a set of parameters called weights whose values can be determined on the basis of a set of examples of the required mapping. The process of determining these parameters values is often called learning or training, and may be a computationally intensive undertaking. Once the weights have been fixed, however, new data can be processed by the network very rapidly.

In a single layer feedforward network, the input layer maps an input vector $x \in \mathbb{R}^d$ to the hidden layer $y \in \mathbb{R}^e$. Then, the output layer maps y to $z \in \mathbb{R}^g$. In this formulation, the input layer maps $x \rightarrow y$ via

$$y = f(Wx + b) \tag{3.6}$$

where $W \in \mathbb{R}^{(e \times d)}$, $b \in \mathbb{R}^e$, and $f(\cdot)$ is an activation function that imposes a non-linearity in the neural network. The hidden layer has a similar formulation:

$$z = f(W_{\text{out}} y + b_{\text{out}}) \tag{3.7}$$

with $W_{\text{out}} \in \mathbb{R}^{(g \times e)}$, $b_{out} \in \mathbb{R}^g$.

The **multilayer perceptron** (MLP) is a type of feedforward neural network that consists of an input layer of neurons, an arbitrary number of hidden layers, and an output layer. A multilayer perceptron acts in much the same way as a single layer feedforward neural network, but with an arbitrary number of hidden layers (and a corresponding number of W's and b's). The multilayer perceptron trains the weights of the W's and b's to minimize some cost function. This cost function should minimize the distance between the output z and some target. The choice of activation functions $f(\cdot)$ and cost functions depends on the domain of a given task.

3.3.1 Stochastic Gradient Descent and Backpropagation

To quote Ruder [2016]:

> Gradient descent is a way to minimize an objective function J(θ) parameterized by a model's parameters $\theta \in \mathbb{R}$ by updating the parameters in the opposite direction of the gradient of the objective function $\Delta_\theta J(\theta)$ w.r.t. to the parameters. The learning rate η determines the size of the steps we take to reach a (local) minimum. In other words, we follow the direction of the slope of the surface created by the objective function downhill until we reach a valley.

In the context presented here, the model parameters θ are the W's and b's of the MLP. Theoretically one can perform gradient descent by passing each input to the network to get an output, compare each output to its target to find $J(\theta)$, calculate this objective function's gradient by summing over all calculated costs, find each parameter's partial contribution to this gradient, and finally update each parameter according to the step size η. This rarely happens in practice, however, due to the computational constraints of performing all these calculations on a large dataset with a large ANN.

Instead, the stochastic gradient descent algorithm is used to approximate this procedure. Rather than summing over all training examples and then updating parameters, stochastic gradient descent updates the ANN parameters incrementally over individual training examples or mini-batches of training examples. Adam [Kingma and Ba, 2015], or its modification AdamW [Loshchilov and Hutter, 2017], are widely used methods for performing stochastic gradient descent with ANNs.

The parameters of an ANN are updated using the backpropagation algorithm. This algorithm can efficiently calculate the contribution of each parameter in an ANN to the cost function for a single input-output pair compared to a naive direct computation of the gradient with respect to each weight individually. The backpropagation algorithm works by computing the gradient of the loss function with respect to each weight by the chain rule, computing the gradient one layer at a time, and iterating backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule. ANN programming libraries such as Tensorflow and Pytorch provide a framework for autodifferentiating these networks [Abadi et al., 2016, Paszke et al., 2019], allowing for the rapid construction, training, and deployment of ANNs.

3.3.2 Cost Functions

In order to train an ANN, a cost function of the model's parameters θ must be chosen. Given some inputs m (and assuming M total inputs) this function measures some distance between the ANN's output $\hat{s}(m)$ and a target $s(m)$. Commonly used cost functions are mean absolute error (MAE)

$$C(\theta) = \frac{1}{M} \sum_{m=0}^{M-1} |s(m) - \hat{s}(m)| \tag{3.8}$$

and mean squared error (MSE)

$$C(\theta) = \frac{1}{M} \sum_{m=0}^{M-1} (s(m) - \hat{s}(m))^2 \tag{3.9}$$

Often times the MAE is referred to as the L1 loss, and MSE referred to as the L2 loss.

In machine learning for audio tasks, another commonly used cost function is the multi-scale spectrogram (MSS) loss. Inspired by the multi-resolution spectral amplitude distance demonstrated in Wang et al. [2020], Engel et al. [2019] defines the MSS as follows:

Assuming $s(m)$ is the target audio and $\hat{s}(m)$ is the estimated audio, compute their magnitude spectrograms S_i and $\hat{S}_i$ with a given FFT size i. The spectrogram loss is defined as the sum of the L1 difference between S_i and $\hat{S}_i$, as well as the L1 difference between $log(S_i)$ and $log(\hat{S}_i)$

$$L_i = ||S_i - \hat{S}_i||_1 + ||log(S_i) - log(\hat{S}_i)||_1 \tag{3.10}$$

The total loss is the sum of all the spectral losses for the various chosen FFT sizes i

$$MSS = \sum^{i} L_i \tag{3.11}$$

Though MAE in the time domain can be used in audio applications, and is cheaper to compute than MSS loss, the latter has recently gained popularity as it can ignore the phase differences between the target and estimated signals. This resilience to slight phase changes is said to mimic human perception [Chi et al., 2005]. Though using several FFT sizes increases the cost and complexity of the loss calculation, Wang et al. [2020] states that "using multiple spectral distances is expected to help the model learn the spectral details of natural waveforms in different spatial and temporal resolutions."

Notes on Interpreting MSS Loss

Table 3.1: MSS loss measured between various signals and their copy -3dB using different sets of FFTs.

	Set A	Set B
Sinewave	1.59	1.53
White Noise	35.72	6.73
Rock Song	0.51	0.13

It is difficult to provide a rule of thumb for interpreting the values of MSS loss. The values MSS loss takes depends on several factors, include the content of the signals being measured, the number of FFTs chosen, and the size of the FFTs. Aside from reaching 0 when comparing two identical signals, it is difficult to intuit even the order of magnitude the MSS loss will take.

Consider the following three 10-second signals: a 1kHz sinewave with amplitude 1; a zero-mean unit-variance white noise signal; and a mono mixdown of the chorus of a rock song ("Haunted House" by the band Woodfire, which is used in Chapter 9). Two sets of FFTs will be used to measure some MSS losses: Set A, which uses sizes $\{4096, 2048, 512\}$, and Set B which uses sizes $\{128, 64, 32\}$. At a 44.1kHz sampling rate, Set A uses FFT windows ranging from 10ms to 93ms, and Set B uses FFT windows ranging from 1ms to 3 ms.

For each set of FFTs, the MSS loss will be measured between each signal and a copy of the signal -3dB. These MSS values using Set A and Set B are presented in Table 3.1.

In general, MSS losses calculated on tonal signals tend to be invariant to changes in FFT sizes. Also, MSS losses tend to take larger values on broadband signals compared to narrowband signals, though this does not necessarily hold across different FFT sizes. Thus it is difficult to gauge, just by looking at a final MSS value, how close of a perceptual match two signals are.

Nevertheless, MSS loss has found success as a cost function in neural network literature. Should researchers want to claim that the output of their algorithm is perceptually indistinguishable from some baseline, it is common practice to run a listening test. See Sec 3.4 for a fuller discussion.

3.3.3 Activation Functions

The following are common activation functions found in ANNs.

Sigmoid Function

The sigmoid function, also known as the logistic function, is commonly found in ANNs:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.12}$$

This function asymptotically approaches 0 as $x \to -\infty$ and asymptotically approaches 1 as $x \to \infty$. As such, this function is often used to continuously describe logical functions.

Hyperbolic Tangent Function

The hyperbolic tangent function (tanh) is also commonly used in ANNs:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.13}$$

This function asymptotically approaches -1 as $x \to -\infty$ and asymptotically approaches 1 as $x \to \infty$. Though this function is similar to the sigmoid function, its gradient can be four times greater than the sigmoids and thus may allow for faster convergence with the backpropagation algorithm. This activation function can also be useful to represent both positive and negative features.

Rectified Linear Unit and Leaky Rectified Linear Unit

The rectified linear unit (ReLU) [Nair and Hinton, 2010]

$$\text{ReLU}(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{3.14}$$

has been used in ANNs to avoid the "vanishing gradient problem [Hochreiter, 1998]," where due to the chain rule an individual neuron's contribution to a gradient calculation can tend towards zero in sufficiently large ANNs with sigmoid or tanh activations. As opposed to the sigmoid or tanh activations, the ReLU's derivative is 0 for negative values and 1 otherwise. Therefore, an ANN composed of ReLU activations will never have a neuron's contribution to the gradient disappear.

However, ReLUs can suffer from the "dying neuron problem [Lu et al., 2019]," in which a poor initialization can lead a neuron's parameters to become static as it is trapped in the negative support of the ReLU where the derivative is zero. Thus the leaky rectified linear unit (LReLU)

$$\mathrm{LReLU}(x) = \begin{cases} \beta x, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{3.15}$$

has been proposed by Xu et al. [2015]. The parameter β is typically chosen to be small (on the order of 10^{-1}) and sets the derivative of its negative support to be nonzero, thus avoiding the dying neuron problem.

3.3.4 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a powerful class of ANNs. CNNs distinguish themselves from fully-connected architectures such as MLPs in a few key ways [Kiranyaz et al., 2021]:

- CNNs combine feature extraction and feature classification into one architecture
- CNNs can efficiently process large inputs due to their sparsely-connected neurons with tied weights
- CNNs can be invariant to small transformations to an input
- CNNs can adapt to different input sizes

For two-dimensional inputs such as audio, the 1D forward propagation of a CNN-layer is expressed as

$$x_k^l = b_k^l + \sum_{i=1}^{N_{l-1}} \mathrm{conv1D}(w_{ik}^{l-1}, s_i^{l-1}) \tag{3.16}$$

where x_k^l is the input, b_k^l the bias of the k^{th} neuron at layer l, s_i^{l-1} is the output of the i^{th} neuron at layer $l-1$, and w_{ik}^{l-1} is the kernel from the i^{th} neuron at layer $l-1$ to the k^{th} neuron at layer l. "conv1D" performs 'in-valid' 1D convolution without zero-padding, thus making the dimension of x_k^l smaller than that of s_i^{l-1}. The intermediate output can be expressed as

$$y_k^l = f(x_k^l) \text{ and } s_k^l = y_k^l \downarrow ss \tag{3.17}$$

where $f(\cdot)$ is some activation function, s_k^l is the output of the k^{th} neuron of the l^{th} layer, and $\downarrow ss$ represents a downsampling operation with factor ss.

Typically the convolution kernels w are thought of as feature extractors that operate at varying resolutions of the input dependent on the downscaling factor ss. Fully connected MLPs can be used after several convolutional layers to produce some embedding of these features and see widespread usage in classification tasks.

3.4 Perceptual Evaluation of Audio Algorithms

Though cost functions are used to fit models and provide a numerical characterization of a system's performance, often a perceptual evaluation is desired for the outputs of an audio algorithm. To quote Bech and Zacharov [2007]

> However, this [numerical] characterisation of the physical audio signal does not tell us how the human auditory system will interpret and quantify it. [...] An alternative means of assessing how listeners perceive an audio signal would be to ask them to quantify their experience, [which] often takes the form of a formal listening test.

In the absence of a sufficiently descriptive numerical measure for an audio signal, a listening test can be performed to answer a researcher's specific question about the outputs of their audio algorithm. Formal listening tests are commonly used to answer questions such as

- Are a set of audio stimuli identical to one another?
- Is an audio stimulus inferior, equivalent, or superior to another stimulus in regards to quality?
- Is an audio system suitable for a given task?

When asking participants of a formal listening test to rate several stimuli simultaneously, e.g. to compare the results of different configurations of an audio algorithm, the Multi Stimulus test with Hidden Reference and Anchor (MUSHRA) is commonly used [BS.1534-3, 2003]. As summarized in De Man et al. [2019], the hallmarks of a MUSHRA test are

- At least 4 stimuli presented simultaneously
- A separate slider per stimulus used to rate some continuous quality scale marked "Bad," "Poor," "Fair," "Good," and "Excellent"
- A reference stimulus is provided which the other stimuli are compared to with attributes including "basic audio quality"
- The reference is also included in the stimuli to be rated as a "hidden reference"
- Among the stimuli to be rated is one or more "anchors," which are meant to present a low-quality comparison

To run a MUSHRA listening test, it is possible to use web versions of the test such as the Web Audio Evaluation Tool or webMUSHRA [Jillings et al., 2015, Schoeffler et al., 2018]. Although distributing a MUSHRA evaluation online does limit a researcher's ability to control the environment and equipment used to perform the test, it greatly expands the pool of potential participants. Web-based MUSHRA listening tests have become a standard in evaluating intelligent music production algorithms and have been used to perceptually evaluate the outputs of all the algorithms cited in Section 3.6.

3.5 Audio Effects Modelling

The following discussion is adapted from Vanhatalo et al. [2022]. While the authors focused on modelling guitar distortion effects, their categorization of audio effect modelling is illuminating.

Depending on the degree of prior knowledge applied to model a target effect, the existing approaches for effect modelling can be divided into three categories: white-, black-, and grey-box. Both traditional signal processing algorithms and neural networks have been applied to these categories.

In the case of greybox modelling, a new family of neural network algorithms referred to as differentiable digital signal processing (DDSP) are foundational to this thesis. Though they technically constitute a branch of the modelling described in Section 3.5.3, they will receive a separate treatment in Section 3.5.4.

3.5.1 Whitebox Modelling

Whitebox modeling is based on the complete knowledge of the system, uses ordinary/partial differential equations to describe its behaviour and adopts numerical methods to solve them in the continuous or discrete domain. Since they reproduce all the important characteristics of a target device, such models can achieve very good results and are preferable when the sound of a specific analog device is to be replicated with high accuracy. Such models can be very time consuming to develop, require exact knowledge of the equations describing nonlinear elements, and can result in substantial computational load. Simple systems can be modelled manually by solving differential equations [Yeh et al., 2007, D'Angelo and Välimäki, 2014, Esqueda et al., 2017]; but, for more complex cases, there exist general-purpose frameworks like: state-space models [Mačák, 2012, Holters and Zölzer, 2015, Yeh et al., 2009], wave digital filters [Werner et al., 2015, 2018, Dunkel et al., 2016, Cauduro Dias de Paiva et al., 2011, De Sanctis and Sarti, 2009, D'Angelo et al., 2012], and port-hamiltonian systems [Falaize and Hélie, 2016].

For artificial reverberation modelling, computational acoustics can be considered a class of whitebox models [Valimaki et al., 2012]. Wave-based methods rely on numerical approximations for solving the wave equation [Savioja et al., 1994], and geometry-based methods assume a ray-like approximation of the propagation of sound waves [Funkhouser et al., 1998]. Whether they directly estimate an IR or estimate time-energy responses, both methods model the underlying physics that govern the reflections of sounds in a room.

Given how mathematically intensive and explicit whitebox approaches tend to be, there are fewer neural network examples compared to blackbox and greybox models. In Parker et al. [2019], the authors adopt a deep neural network in the context of a state-space model for distortion modelling, which they call state trajectory network. The network uses both the input signal and an internal state to predict the output. The authors apply the method to a first-order and a second-order diode clipper.

In Esqueda et al. [2021] the authors introduce the concept of differentiable whitebox virtual analog modeling, with the idea of using backpropagation to optimize the components' values in an analog distortion circuit. The authors apply this method to find the resistors' and capacitors' values that best approximate the frequency response of an RC filter and a tone-stack.

Another recent work, Parker et al. [2022], uses recurrent neural networks with fast convolutional layers to model partial-differential equations governed systems, focusing on the modelling of acoustic, mechanical and electrical systems. Specifically, they use the proposed approach to investigate a lossy dispersive string, a 2D wave equation, and a tension modulated string.

3.5.2 Blackbox Modelling

Blackbox modeling requires no prior knowledge about the system and relies exclusively on input-output measurements. The main advantage of such approaches is that they simplify the modeling procedure to gathering sufficient data, but they might require time-consuming optimisations, and they seldom offer any interpretability.

Non-neural network blackbox algorithms that have seen applications in effects modelling are the Volterra series and dynamic convolution. The mathematical complexities of Volterra series and dynamic convolution fall outside the scope of this discussion, but heuristically they can be thought of as higher order impulse responses to nonlinear systems. Because of their ability to model systems with memory, Volterra series have seen application in analogue circuit modelling. In Hélie [2006] Volterra series were used to emulate a Moog ladder filter, and in Orcioni et al. [2018] they were used to model tube distortion effects. Though dynamic convolution algorithms have weaker memory modelling capabilities, their explicit modelling of system behaviour at different input levels have seen them used for guitar pre-amp circuit modelling [Primavera et al., 2012b,a].

Neural networks have assumed the lion's share of new approaches to blackbox audio effect modelling over the past few years. Neural networks are a natural fit for blackbox modelling as ANNs learn behaviour strictly from input and output pairs. In Martínez Ramírez [2021], the author presents a generalized neural network approach to effects modelling. With minor tweaks to the neural network topology, the author demonstrates that an ANN can perform (separate) blackbox modelling of EQ, distortion, guitar amplifiers, (multiband) dynamic range compression, chorus, flanger, phaser, tremolo, vibrato, auto-wah, ring modulator and a Leslie speaker. Each of the presented ANNs is comprised of an analysis front-end (which extracts features of the audio input to the effect) that passes information to a deep neural network latent space, whose latent representation of the input audio drives a synthesis back-end that modulates the input audio with adaptive filters. One drawback of this approach, however, is that the ANNs are only trained on one set of parameters for each effect. For example, one ANN must be trained to emulate a lowpass EQ while another must be trained to emulate a highpass EQ.

Blackbox networks have also been used to model DRC effects. In Hawley et al. [2019], a joint real/imaginary neural network is trained on input/output spectrogram pairs to model an LA-2A compressor. Steinmetz and Reiss [2022] also models the LA-2A compressor, using time-based convolutional layers known as TCNs. In both cases, the blackbox algorithms learn to apply dynamic range compression using neural network based operations rather than the DSP calculations shown in Section 2.6. Furthermore, both blackbox algorithms are limited by the parameterization of the LA-2A; the blackbox algorithms learn DRC parameterized using the LA-2A's peak reduction knob and the compress/limit switch parameters (rather than all the parameters outlined in 2.6). While this may not be a bad outcome, it does underscore the fact that blackbox algorithms are limited to the data on which they are trained.

The blackbox network proposed in Steinmetz [2020] is able to model several effects simultaneously, as well as have the parameter settings of those effects exposed and changeable. This network simultaneously models gain, panning, EQ (in the form of a 5 band graphic EQ), DRC, and artificial reverberation. By training on input/output audio pairs as well as the parameter settings for each of the effects, the network avoids the need to be re-trained for each effect. Moreover the parameters sent to the neural network are chosen such that they are familiar to musicians and mix engineers, e.g. the attenuations in each band of the graphic EQ, the threshold of the DRC, or the decay of the artificial reverberation.

3.5.3 Greybox Modelling

Greybox approaches combine a partial theoretical structure, referred to as block-oriented model, with data – typically input/output measurements – to complete the model. Greybox models have the advantage of greatly reducing the prior knowledge necessary to model a device while maintaining a degree of interpretability, thanks to the block oriented approach. The specific structure – together with the measurement and optimization procedures – are critical to achieve a good approximation, especially for nonlinear systems where the output is a function of the input signal amplitude.

These models are typically represented as an interconnection of linear filters and static nonlinearities, such as: Hammerstein models (static nonlinearity followed by linear filter), Wiener models (linear filter followed by static nonlinearity) or Wiener-Hammerstein models (static nonlinearity inbetween two linear filters) [Novak et al., 2009, 2016, Cauduro Dias de Paiva et al., 2012, Rébillat et al., 2010]. But they also include more complex arrangements like cascaded and parallel blocks, such as those presented by Schoukens and Ljung [2019].

In the case of distortion circuits and amplifiers, Wiener-Hammerstein models have been extended to include: non-static nonlinearities (i.e. hysteresis and memory) [Eichas et al., 2015, Eichas and Zölzer, 2016], and pre- and power-amp modeling [Kemper, 2014, Eichas et al., 2017, Eichas and Zölzer, 2018].

For artificial reverberation, feedback delay networks can be considered a family of greybox models [Smith, accessed 03 November, 2022]. Feedback delay networks approximate the reflections of natural reverberation by sending input audio through parallel delay lines with absorption equalisers in the forward pass, a dispersion matrix in the feedback connection, and a tone equaliser right before the network's output. These greybox models have found usage in machine learning in reverberation matching tasks, and are often optimized using genetic algorithms [Ibnyahya and Reiss, 2022, Coggin and Pirkle, 2016, Chemistruck et al., 2012].

3.5.4 Differentiable Digital Signal Processing

The term "differentiable digital signal processing" (DDSP) was proposed by Engel et al. [2019], in which common DSP modules are manually implemented in a differentiable framework such as Tensorflow or Pytorch. This autodifferentiated regime allows for these modules to be implemented in or controlled by neural networks due to their ability to backpropagate gradients. In Engel et al. [2019], an ANN was proposed that used differentiable harmonic oscillators, noise filtered through FIR EQs, and convolutional reverb to synthesize audio. Since then, many individual audio effects have been implemented using the DDSP paradigm.

Nercessian [2020] presents a DDSP inspired parametric EQ. The authors train an ANN that, given an arbitrary magnitude response, estimates the parameters of a parametric EQ that matches said response. The parametric EQ is comprised of a low shelf filter, high shelf filter, and 10 peaking filters. The ANN learns to control the cutoff filter, gain, and Q of each of these biquads. By implementing complex polynomial evaluation in an autodifferentiating framework, the author enabled an ANN to directly control a parametric EQ via the frequency sampling method rather than controlling a proxy blackbox network.

This parametric EQ formed the basis for a DDSP distortion algorithm presented in Nercessian et al. [2021]. This network learns to match a distortion effect by controlling a cascade of parametric EQs and tanh nonlinearities. The values of these EQs and tanh blocks are "hyperconditioned," which means information regarding the parameter settings of the distortion effect are injected into the network. The authors baseline their approach by modelling a BOSS MT-2 distortion pedal, ultimately concluding that the DDSP distortion competed reasonably well against a blackbox neural network model that had a hundred times more parameters.

Parametric reverberation was implemented using DDSP in Lee et al. [2022]. The authors present three different types of DDSP artificial reverberators: a filtered velvet noise effect, an advanced filtered velvet noise effect, and a feedback delay network. Using a similar framework as Nercessian [2020], the authors implement the various FIR and IIR EQs that appear in traditional artificial reverberators using DDSP. The authors baseline the performance of their algorithms on two tasks: an analysis synthesis task, and a blind estimation task. As opposed to the blackbox reverb estimation algorithm presented in Sarroff and Michaels [2020], this DDSP algorithm produces a fully characterized RIR whose color and tone can be tweaked by modifying its learned EQs.

While blackbox neural networks have been used to emulate DRCs [Hawley et al., 2019, Steinmetz and Reiss, 2022], to the author's knowledge only two DDSP DRCs has been proposed in the literature: Steinmetz et al. [2022] and Wright et al. [2022]. In Steinmetz et al. [2022] the authors implemented a DRC using Pytorch with tunable threshold, ratio, knee width, makeup gain, and a ballistics control. The authors approximate both attack and release time with a joint smoothing parameter that controls a single pole IIR filter to smooth the DRC's attenuation curve. This IIR filter is then approximated using an FIR filter. As stated by the authors, forcing the attack time and release time to be shared restricts the modelling capabilities of the DRC. The work presented in Wright et al. [2022] applies a similar technique, but adds complexity to the one-pole filter used for smoothing and can separately model attack and release.

Outside of individual effects modelling, DDSP has also been used in synthesis tasks. In Hayes et al. [2021] the authors present a waveshaping synthesizer. Using a similar architecture to Engel et al. [2019], the authors train neural networks to learn a bank of waveshapers, which are then used to synthesize audio. Another algorithm, presented in Shan et al. [2022], uses a DDSP wavetable synthesizer capable of learning a bank of periodic wavetables. Most recently a differentiable DX7 synthesizer was presented in Caspe et al. [2022], which implemented FM synthesis using DDSP techniques.

Similarly, singing voice synthesis has also seen implementations using DDSP, such as the algorithms presented in Nercessian [2021] and Nercessian [2022]. While Nercessian [2021] uses the architecture shown in Engel et al. [2019] to synthesize singing voices, Nercessian [2022] presents a novel DDSP implementation of the WORLD vocal synthesizer [Morise et al., 2016].

DDSP presents an exciting new direction in machine learning for audio, as it leverages the wisdom of traditional DSP algorithms with the modelling capacity of neural networks. Interpretability of module parameters is commonly found in DDSP approaches as they often seek to update and improve on well established, traditional algorithms for audio effects and synthesis. As such, this dissertation will use DDSP as a guiding paradigm in the development of legible modules for use in IMP tasks such as reverse engineering a mix.

3.6 Automatic Multitrack Mixing with Neural Networks

Several publications have proposed ANNs to automate the mixing of multitracks. These papers outline architectures that use "end-to-end" methods, which means the input and output to the networks are strictly time-domain audio. The networks do not take intermediate representations for input or output, such as the feature embeddings described in Section 3.2, and instead learn to complete some task using only audio. In terms of a mixing task, an end-to-end system would be able to learn about mix engineering using raw tracks and associated mixdowns.

3.6.1 Intelligent Drum Mixing with the Wave-U-Net

In Martínez Ramírez et al. [2021], the authors present an ANN that can mix a batch of raw drum tracks to a stereo mixdown . The architecture is based on the "Wave-U-Net," which was originally developed for a source separation task Stoller et al. [2018].

The raw audio of K stems are input to the neural network on the left side and passed through downsampling blocks. These blocks apply 1D convolutions and then perform a down-sampling operation. This serves to create increasingly small and abstract feature representations of the input multitrack. After performing L downsampling operations, the network then begins to upsample the feature representations as shown on the right of the diagram. Corresponding representations in the downsampling blocks are cropped and passed to the up-sampling blocks, forming skip connections. Thus the network is able to retain the feature map of a downsampling block past the network's bottleneck and use it to inform the upsampling operation. Finally the network outputs two channels of audio, which are meant to be the left and right channels of a stereo mix of the K input stems.

The authors chose to use the ENST drums dataset for their work [Gillet and Richard, 2006]. The dataset consists of several hours of solo drum performance recorded on eight channels and mixed to stereo. The performances include individual hits, fills, solos, and accompanied playing. The eight channels each correspond to a mic on a part of the drum kit, i.e. kick, snare, tom 1, etc. Implicit in the dataset is a notion of how to mix a set of eight raw tracks into a stereo mixdown, based on what content is present in each track of the multitrack and the desires of the mix engineer who mixed the dataset.

To train the network, a 2.75 second snippet of each time-aligned stem acts as input and 2 second snippets of the left and right channels of the stereo mixdown (that corresponds to the center part of the inputs) act as targets. This gives the network a "receptive field" and allows the network to make mixing decisions based on audio that occurs right before and right after the target 2 second snippet of the multitrack. An L1 loss is calculated between the output of the network and the target mixdown for both the left and right channels. All audio is sampled at 44.1 kHz.

Given this training setup, the network is tasked with learning when and how to apply audio processing to the stems in order to produce a stereo mixdown. Because the mixdowns in the ENST dataset use gain, panning, EQ, reverb, DRC, and distortion, the network is tasked with learning how to apply these effects. Furthermore, the network must learn when to apply effects given what audio is presented at the input. It is important to note that the network can only mix drums with this setup, and that the drums must always be input to the same channels as they were during training. For example, if the kick drum was placed as stem 2 during training, then it must be placed as stem 2 for any inference.

3.6.2 Differentiable mixing console of neural audio effects

In Steinmetz et al. [2020], the authors present an ANN "differentiable mixing console." Each raw track is passed to an encoder network, which passes an embedded representation of each input track to a context processor. This unit outputs the parameters that control the transformation network. This transformation network is tasked with applying processing such as gain, panning, EQ, reverb, and DRC based on the parameters from the post-processor. The transformation network then applies effects to each input track, and the results are summed to make a left and right channel of a mixdown. A spectrogram based stereo loss function is used as the cost function in lieu of an L1 penalty.

The network is set up such that across N inputs, the encoder weights, post-processor weights, and transformation network weights are shared. In other words, there are N copies of the same neural network that each individually process the N input tracks. This allows for flexibility with the order in which the raw tracks are fed to the model. Furthermore, the encoder network is a spectrogram-based VGGish architecture which is able to generate embeddings for a wide domain of input audio, rather than just being trained on drums [Hershey et al., 2017].

A stereo-invariant multi-scale spectrogram cost function is proposed by the authors, as it was found that the network could not learn to pan properly when trained on a multi-scale spectrogram loss on the mixdowns' left and right channels.

3.6.3 Automatic music mixing with deep learning and out-of-domain data

Martínez-Ramírez et al. [2022] present an ANN that can mix a multitrack based on a novel methodology for preprocessing the multitrack.

As the authors note, one difficulty when training an ANN to mix multitracks is the relative lack of paired clean raw multitracks and mixdowns. They go on to note that a wealth of wet stems and mixdowns are available through source-separation tasks. In a vacuum, however, these wet stems cannot be used to train an ANN mixing system as the mixdown is often an unweighted sum of the stems.

As such, the authors present a methodology of measuring the various effects applied to wet stems in these datasets. Instead of removing the audio effects found on these stems before mixing them, the authors normalize each stem based on audio features related to several classes of audio effects. These normalizations consider loudness, equalisation, panning, dynamic range compression, and reverberation. An ANN is then trained to mix these effect-normalized stems to their associated target mixdown.

With this preprocessing methodology established, the authors then demonstrate how to mix a multitrack of raw tracks. Each K raw track is effect normalized according to the distribution of effects mentioned in the previous paragraph. Then, this raw track is passed to the ANN's adaptive front end, which extracts salient features of the raw track. These features are then passed to the ANN's latent space mixer, which learns a mixing mask to apply to the raw track in the ANN's synthesis backend.

The authors present a stereo-invariant loss function similar to Steinmetz et al. [2020].

3.7 Reverse Engineering Mixes

Gorlow and Marchand [2013] provides a neat conceptual encapsulation of reverse engineering in audio:

> The objective of reverse audio engineering can be either to identify the transformation parameters given the input and output signals [...] or to recover the input signal that belongs to the output signal given the transformation parameters, or both. An explicit signal and system model is mandatory in either case.

One very large branch of this field is source separation [Makino, 2018]. As its name suggests, source separation involves separating the individual sources that make up an audio mixture. While a full literature review of this field falls outside the scope of this dissertation, it is worthwhile to mention due to its sheer size and influence on machine learning for audio.

What follows are a few examples of reverse engineering literature that adhere more closely to the definition provided above.

3.7.1 Reverse Engineering DJ Mixes

Schwarz and Fourer [2021] presents a methodology and dataset for reverse engineering DJ mixes. As with mix engineering, several machine learning based approaches have emulated the behaviours of DJs, such as the neural network presented in Chen et al. [2022] and the MIR driven approach shown in Kim et al. [2017]. However, Schwarz and Fourer [2021]'s approach is concerned with extracting information from human-made mixes rather than seeking to automate their choices. According to the authors, in order to automatically annotate DJ mixes the following components are required:

- Identification of tracks used in the DJ mix
- Alignment of where the used tracks start and stop within the mix
- Time-scaling to match speech changes that may have been used by the DJ
- Unmixing to estimate the cue regions where the cross-fades between tracks happen, the curves for volume, bass and treble, and the parameters of other effects

- Content and metadata to inform about the choices a DJ makes when creating a mix

The authors choose to focus on the alignment, time-scaling, and unmixing components of this problem. This is different to another DJ mix unmixing work presented in Ramona and Richard [2011], which does not consider time-scaling. Their method begins with a rough alignment of the tracks identified in the mix, which is then refined to sample precision, after which gain curves and cue regions are estimated.

The authors also present a dataset for use in their reverse engineering task referred to as the UnmixDB Dataset [Schwarz and Fourer, 2018]. This dataset consists of 444 mixes, each made by blending three tracks from a set of 37 total tracks. One of four effects (bypass, bass boost, distortion, or DRC) are applied to each track, as well as one of three time-altering algorithms (bypass, resampling, or stretching).

The authors evaluate their reverse engineering method using objective measures of frame, sample, and suppression error for alignment, speed error for time-scaling, and fade error for both. These error statistics are measured accross the whole dataset, and are also broken down into different combinations of effects applications between blends to measure their sensitivity to different effects processing. The authors conclude by mentioning “with some refinements, our method could become robust enough to allow the inversion of fading, EQ, and other processing.”

Though the UnmixDB mixes are not as natural as those presented in other works such as Sonnleitner et al. [2016], the authors maintain that access to ground truth labels regarding cue points and time altering are crucial to the evaluation of their reverse engineering approach. Furthermore, the authors do not attempt to estimate the effects processing applied to the tracks in a blend.

3.7.2 Reverse Engineering Two Stage Cascade Mixes

Gorlow and Marchand [2013] reduces mixing and mastering to a simplified process in order to solve a reverse engineering and source separation problem:

> Given the mixing, mastering, and signal parameters listed, recover the source signals from a mixture signal with a compressed dynamic range in the best possible quality.

In this formulation, the authors reduce multitrack mixing to gain and panning, and reduce mastering to dynamic range compression. The authors seek to estimate the parameters for panning angle, gain power, compression detection type, threshold, compression ratio, attack/release times, and makeup gain. After the estimation of these parameters, the authors attempt to recover the original, uncompressed signals present in the mixture.

Their method is evaluated using one mix of a five-track multitrack that is 24 seconds in length. Objective measures of the signal recovery include an RMS error between the ground truth and estimated signals, as well as a perceptual similarity measure known as PEMO-Q [Huber and Kollmeier, 2006]. The authors use the PEMO-Q measure in lieu of a formal listening test. According to Huber and Kollmeier [2006], “to evaluate the audio quality of a given distorted signal relative to a corresponding high-quality reference signal, the auditory model is employed to compute ”internal representations” of the signals, which are partly assimilated in order to account for assumed cognitive aspects." These mathematically based perceptual measures find widespread usage when evaluating speech and audio codecs.

3.7.3 The Reverse Engineer of a Mix

Barchiesi and Reiss [2010] serves as the foundation of this dissertation. In this paper are

> two algorithms based on a least-squares optimization that can be used for reverse engineering a mix. The evaluation of our techniques shows that, given the raw multitrack recording and the final or target mix, it is possible to estimate the parameters of a wide range of different effects, including linear time-invariant processors (gains, delays, stereo panners, and filters) and dynamic effects.

The authors note that they present an improvement to Kolasinski [2008], which employed a genetic algorithm to estimate the gains used to mix a multitrack down to a mono mixdown.

A thorough description of the authors' proposed linear effect estimation is presented in Chapter 4. To summarize, the authors use a least-squares estimation procedure based on linear algebra that can recover the magnitudes of taps in an LTI filter used to mix a multitrack. The authors separately use this MIMO least-squares estimation procedure to estimate gain envelopes that can model dynamics processing in multitrack mixing. To allow for gain variation within a frame and to encourage smoothness in the envelope estimation, the authors model the gain in each frame as a polynomial of a specified order. In their final evaluation, the authors chose a polynomial of order 4 and a frame size of 128 samples.

To evaluate the linear processing, the authors mix a 30-second long four track multitrack by applying gain, delay, FIR EQ, IIR EQ, and panning. By setting their FIR filter estimation order to 100, the authors are able to recover each IR used in their mixing. The authors also present a "real-world" reverse engineering demonstration, where a six-track recording was mixed using Apple Logic Pro software using only linear processing. Ultimately the algorithm is able to fit order 512 FIR EQs to each track in the multitrack, resulting in a MSE of 5.42×10^{-4} between the target and estimated mix. The authors note that the EQ matching of the algorithm performs better in frequencies below 10kHz due to quantization noise in the target mix.

To evaluate the dynamics processing, the authors mixed an eight track multitrack using DRC on each track with random settings for threshold, compression ratio, attack time, and release time. Three different compressor models with different gain computers were used to measure the modelling power of the polynomial gain estimation. The authors state that the reverse engineering algorithm almost exactly estimates the compression envelopes in two of the three test cases, with the model slightly underperforming in the case of a DRC with the least smooth envelopes. Graphs are shown overlaying the ground truth compression envelope with the learned envelope, but no objective measures are presented.

Chapter 4

Time-Invariant FIR Audio Effect Matching

This section outlines procedures for matching time-invariant FIR audio effects given an input signal $x[n]$ and output $y[n]$. Methods are described to match gain, panning, FIR EQ, and convolutional reverb.

Mathematically speaking, gain, panning, FIR EQ, and convolutional reverb (which includes delay and echo) all constitute FIR LTI systems. Given the formulation above, where both the input and output signal are assumed available, a closed form time domain solution exists that is outlined in Barchiesi and Reiss [2010].

However, computational constraints prevent using the time domain solution for matching convolutional reverb with a sufficiently long IR. In this case, spectral methods are considered. A classical frequency division method is compared to a newer gradient descent method enabled by Engel et al. [2019].

Therefore, Sections 4.1 and 4.2 constitute a literature review, and Section 4.3 presents a measure of matching performance and comparison to a classical method not explicitly demonstrated in literature (i.e. [Engel et al., 2019]).

4.1 Gain and Pan Matching

Given an input signal $x[n]$ that has had some nonzero gain g applied to produce an output $y[n]$, it is possible to recover g via

$$g = \frac{y[n]}{x[n]} \tag{4.1}$$

In the multitrack case, Barchiesi and Reiss [2010] proposes a closed-form solution to match a set of gains g_i applied to the tracks $x_i[n]$ of an N track multitrack to produce a mono output $y[n]$

$$y[n] = \sum_{i=0}^{N-1} g_i x_i[n] \tag{4.2}$$

This linear combination can be recast as the matrix operation

$$y = \boldsymbol{X}g \tag{4.3}$$

where $\boldsymbol{X}$ is a matrix whose ith column is $x_i[n]$ and g is a column vector whose ith entry is g_i. Assuming that $\boldsymbol{X}$ is not singular, an optimal set of coefficients $\hat{g}$ can be found that minimize the Euclidean distance (i.e. MSE) $||y - \boldsymbol{X}g||$ via the least squares formula

$$\hat{g} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T y \tag{4.4}$$

In the case that $\boldsymbol{X}$ is singular, $\boldsymbol{X}^{-1}$ does not exist and thus this method cannot be used. Singularity may arise when one track in the multitrack is muted (i.e. all zeros) or when one track in the multitrack is a subgrouping of other tracks in the multitrack (i.e. a linear combination of tracks in the multitrack).

The same procedure can be used to match the gan and pan within a stereo mix, producing a set of gains g_L and g_R that match the left channel $y_L[n]$ and right channel $y_R[n]$ respectively. These left and right channel gains can be factored into gain and pan parameters (p_L and p_R) using the -3dB panning law described in Section 2.2:

$$\sqrt{g_L^2 + g_R^2} = \sqrt{g^2p_L^2 + g^2p_R^2} = \sqrt{g^2(p_L^2 + p_R^2)} = g \tag{4.5}$$

$$p_L = \frac{g_L}{g}, \quad p_L = cos(\theta) \implies \theta = arccos(p_L) \tag{4.6}$$

$$p_R = \frac{g_R}{g}, \quad p_R = sin(\theta) \implies \theta = arcsin(p_R) \tag{4.7}$$

In this formulation, an additional polarity parameter would need to be included should the signs of g_L and g_R not match.

4.2 EQ Matching

Assume that $y[n] = x[n] * h[n]$, i.e. that $y[n]$ is the result of applying some EQ $h[n]$ to $x[n]$. Mathematically speaking it is possible to recover $h[n]$ using $x[n]$ and $y[n]$ by calculating an inverse filter:

$$y[n] = x[n] * h[n] \implies Y[\omega] = X[\omega]H[\omega] \implies H[\omega] = \frac{Y[\omega]}{X[\omega]} \tag{4.8}$$

However, there is no guarantee that this filter is stable. Moreover, in the absence of information regarding the length of $h[n]$ or in the event of truncation of $y[n]$ this method cannot guarantee a perfect match. As such, methods for EQ matching differ based on whether the EQ effect being fit is FIR or IIR.

4.2.1 FIR EQ Matching

Assuming a Pth order LTI system it is possible to estimate $h[n]$ using the frequency sampling method mentioned in Section 2.3.1. By taking a length P DFT of $y[n]$ and $x[n]$, Equation 4.8

suggests

$$H[\omega] = \frac{Y[\omega]}{X[\omega]} \implies |H[\omega]| = \frac{|Y[\omega]|}{|X[\omega]|} \tag{4.9}$$

Thus by dividing the magnitude responses of $y[n]$ and $x[n]$ and inverting to the time domain, $h[n]$ can be estimated.

Barchiesi and Reiss [2010] point out that a formulation similar to that presented in Sec 4.1 can be used to reverse engineer FIR EQs applied to a single track or a multitrack. Assuming a Pth order LTI system, the single track formulation $y[n] = x[n] * h[n]$ can be recast as

$$y = \boldsymbol{X}_p h \tag{4.10}$$

where $\boldsymbol{X}_p$ is a matrix whose ith columns contains a copy of $x[n]$ that has been shifted (i.e. zero-padded) i samples. In the multitrack case $\boldsymbol{X}_p$ is replaced with $\boldsymbol{X}_{K,p}$, which contains shifted versions of each of the k tracks in the multitrack.

4.3 Reverb Matching

Using the same conventions as Section 4.2, assume that $y[n] = x[n] * h[n]$, i.e. that $y[n]$ is the result of applying some reverb IR $h[n]$ to $x[n]$. Mathematically speaking, the LSE solution proposed by Barchiesi and Reiss [2010] ought to work when trying to match a reverb IR. In practice, however, this approach is not viable due to computational constraints.

The matrix inversion, which is the most expensive computation in Equation 4.4, scales $\mathcal{O}(n^3)$ in time, with n being of taps in the LTI system. Case in point, to reverse engineer a one second convolutional reverb at CD quality, the time domain LSE algorithm requires a transpose, multiplication, and inversion of a 44100x44100 matrix. This matrix contains about 2 billion values, which is too large for most modern hardware (e.g. this 44100x44100 matrix of random values would not fit on the author's laptop with with 16GB RAM).

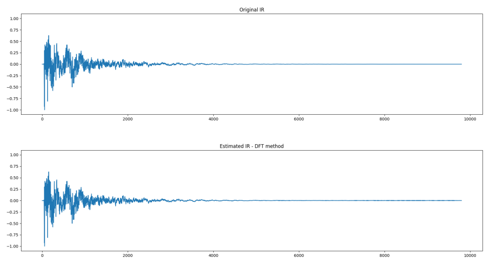
The reverb's IR can be retrieved using frequency-domain approaches. Two approaches will be discussed: using explicit division in the frequency domain; and using stochastic gradient descent with a frequency domain based reverb application.

4.3.1 Case 1: No truncation

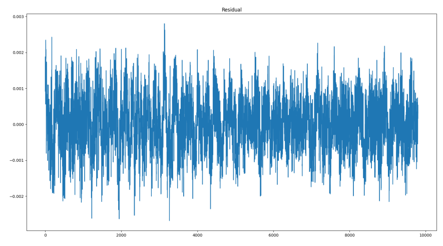
Assume the dry signal $x[n]$ has support of length M and the wet signal $y[n]$ has support of length $M + N - 1$ for some $N \geq 1$. In this case it can be deduced that the reverb IR $h[n]$ has length N due to the nature of convolution.

An approximation of the reverb IR can be recovered using division in the frequency domain using the following procedure. First zero-pad $x[n]$ to match the length of $y[n]$, then multiply each signal with a Hann window of length $M + N - 1$, and then take the DFT of each to find $X(\omega)$ and $Y(\omega)$. From there, divide $X(\omega)$ from $Y(\omega)$ and perform the IDFT on the quotient. This yields a time-domain signal with length $M + N - 1$ where the first N samples are the estimate of $h[n]$.

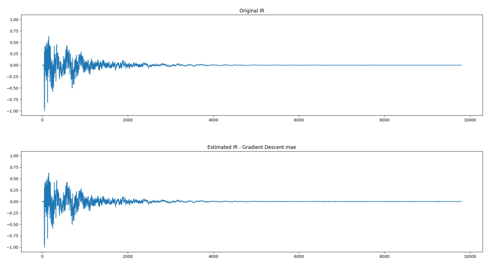
It is also possible to estimate $h[n]$ using gradient descent. First, a length N vector $h[n]$ is initialized with Gaussian noise that has mean 0 and variance 10^{-3}. Then, this reverb IR is



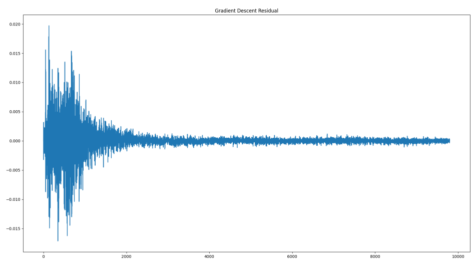
(a) Frequency Division Estimate



(b) Frequency Division Residuals



(c) MAE Gradient Descent Estimate



(d) MAE Gradient Descent Residuals

Figure 4.1: Various reverb IR estimates in Case 1

applied to $x[n]$ using a multiplication in the frequency domain to produce an estimated $\hat{y}[n]$. Finally a MAE is calculated between $y[n]$ and $\hat{y}[n]$ and used as the cost function to update the values of $h[n]$. In this procedure, the gradient descent begins with a learning rate of 10^{-4} until early stopping, then reduced to 10^{-5} until early stopping, and finally reduced to 10^{-6} until early stopping is reached.

The result of each procedure can be seen in Figure 4.1. Both methods do a good job of matching the overall shape of the original IR. Note that the residuals for the frequency division method are on the order of 10^{-3}, whereas the residuals of the gradient descent method are on the order of 10^{-2}. Moreover, the residuals from the frequency division method are more uniformly distributed than those of the gradient descent method, which closely mirror the shape of the original IR.

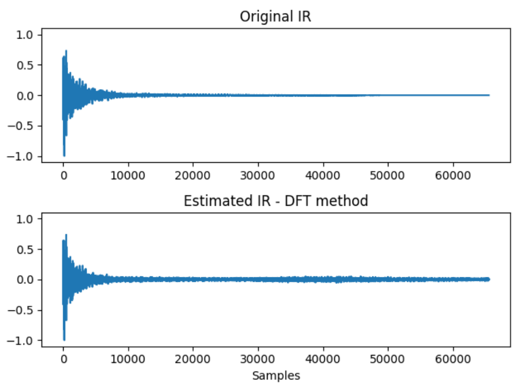
4.3.2 Case 2: Truncation

Assume the dry signal $x[n]$ and wet signal $y[n]$ both have support of length M. In this case no information is immediately available regarding the length of the reverb IR $h[n]$. A situation such as this can arise when dealing with an exerpt of a mixdown or bounce. Though many algorithms deal with the estimation of reverb parameters in a blind setting [Scharrer and Vorländer, 2010, Löllmann et al., 2010, Sarroff and Michaels, 2020], they fall outside the scope of this dissertation. One can estimate an $h[n]$ of a desired length $N \leq M$ using either the frequency division method or using gradient descent.

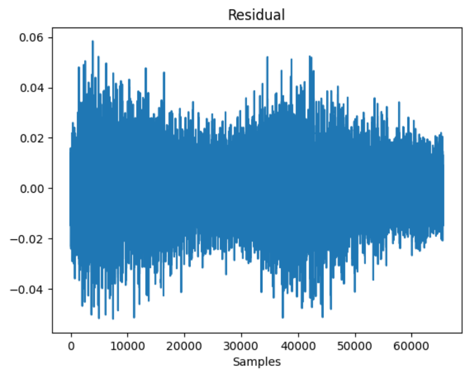
To estimate $h[n]$ using the frequency division method, first truncate $x[n]$ and $y[n]$ to length N, then multiply each signal with a Hann window of length N, and then take the DFT of each to find $X(\omega)$ and $Y(\omega)$. From there, divide $X(\omega)$ from $Y(\omega)$ and perform the IDFT on the quotient. This yields a time-domain signal with length N, the estimate of $h[n]$.

It is also possible to estimate $h[n]$ using gradient descent. First, a length N vector $h[n]$ is initialized with Gaussian noise that has mean 0 and variance 10^{-3}. Then, this reverb IR is applied to the entirety of $x[n]$ using a multiplication in the frequency domain to produce an estimated $\hat{y}[n]$. Finally a MSE is calculated between $y[n]$ and $\hat{y}[n]$ and used as the cost function to update the values of $h[n]$. In this procedure, the gradient descent begins with a learning rate of 10^{-4} until early stopping, then reduced to 10^{-5} until early stopping, and finally reduced to 10^{-6} until early stopping is reached.

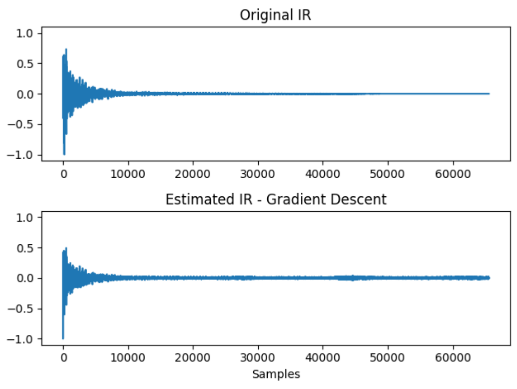
The result of each procedure can be seen in Figure 4.2. Both methods do a good job of matching the overall shape of the original IR. Note that the residuals for the frequency division method are on the order of 10^{-3}, whereas the residuals of the gradient descent method are on the order of 10^{-2}. Moreover, the residuals from the frequency division method are more uniformly distributed than those of the gradient descent method, which closely mirror the shape of the original IR.



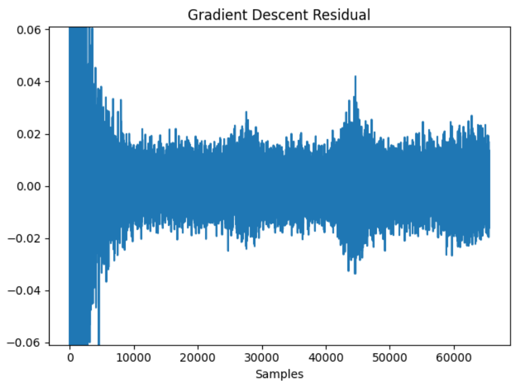
(a) Frequency Division Estimate



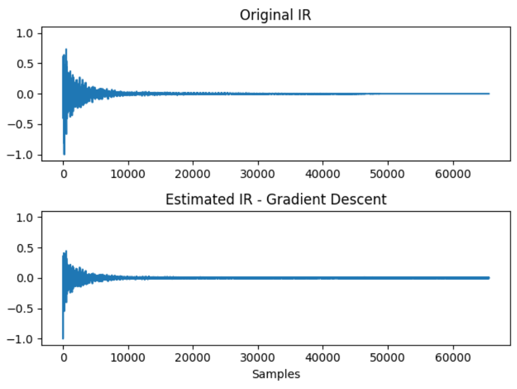
(b) Frequency Division Residuals



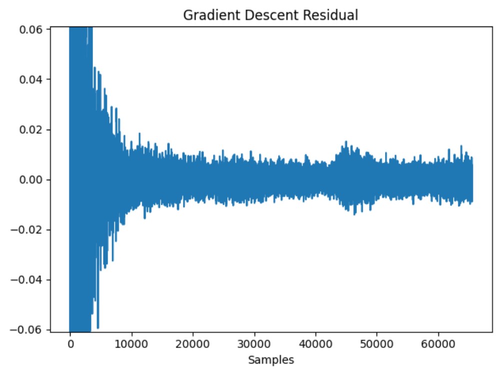
(c) MSE Gradient Descent Estimate



(d) MSE Gradient Descent Residuals



(e) MAE Gradient Descent Estimate



(f) MAE Gradient Descent Residuals

Figure 4.2: Various reverb IR estimates in Case 2

Chapter 5

Time-Invariant IIR Audio Effect Matching

Continuing from Section 4.2, assume that some magnitude response has been calculated using Eqn 4.9 and it is desired to match an IIR filter to this magnitude response.

Classical methods for designing an IIR filter that achieves an arbitrary magnitude and/or phase response include the modified Yule-Walker (MYW) estimation [Chan and Langford, 1982], least squares approaches [Kobayashi and Imai, 1990, Lang, 1998], linear programming [Rabiner et al., 1974], Steiglitz-McBride [Stoica and Soderstrom, 1981], and gradient-based optimization methods [Dodds, 2020].

However, these approaches have drawbacks that may limit their application in scenarios that require high accuracy, fast estimation, or both. For example, while MYW can be performed quickly with a small number of operations, it may produce inaccurate results for more challenging target responses. On the other hand, iterative methods often provide greater accuracy and can be tailored with customized loss functions. However, this comes with higher computational cost due to need for multiple gradient update operations. In addition, since this optimization process is generally non-convex, performance is often very sensitive to initial conditions and also may suffer from getting stuck in local minima [Dodds, 2020, Nercessian, 2020].

Recently, there has been interest in integrating deep learning approaches for filter design. The parallels between recurrent neural networks (RNNs) and IIR filters have been exploited to learn arbitrary filters from data [Kuznetsov et al., 2020, Pepe et al., 2020, Ramírez and Reiss, 2018]. While these networks simulate the sample-by-sample operations of a digital IIR filter, they can be slow and difficult to train due to their recursive nature, which requires many gradient steps through time.

Other approaches have instead been trained to directly estimate the parameters of graphic [Välimäki and Rämö, 2019] and parametric equalizers [Nercessian, 2020, Yospanya et al., 2021] given a desired magnitude response. While these approaches avoid the need for iterative estimation, they are potentially restricted by the IIR filter prototypes they estimate.

To address these limitations, a novel MLP can be constructed that is capable of learning the mapping from a desired arbitrary magnitude response to the coefficients of an IIR filter,

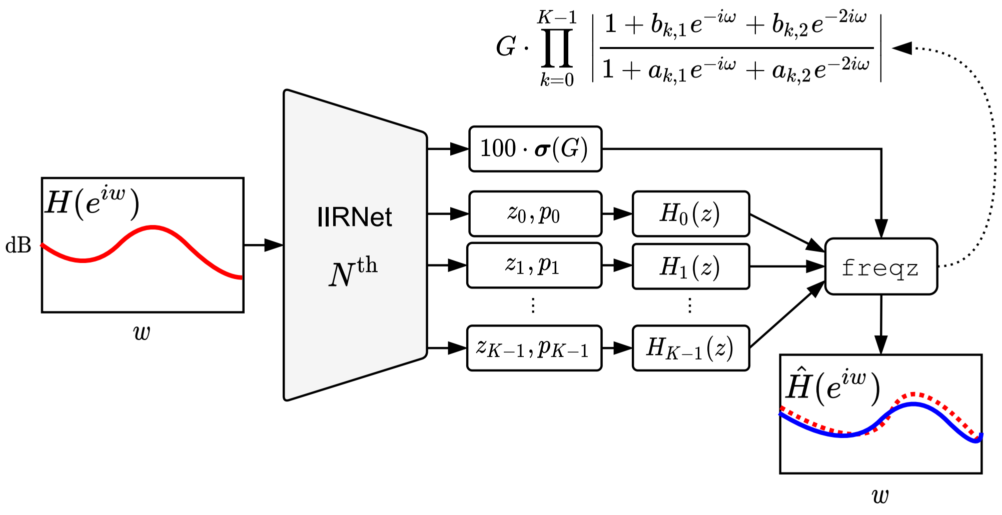


Figure 5.1: Block diagram of IIRNet.

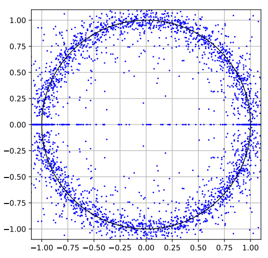
removing the need for iterative optimization. This MLP, named "IIRNet," is trained with randomly generated filters to estimate a cascade of biquads given a desired magnitude response, as shown in Figure 5.1.

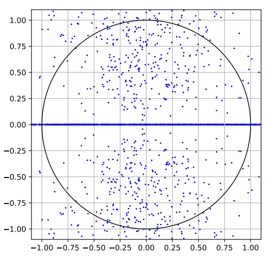
This chapter is organized as follows:

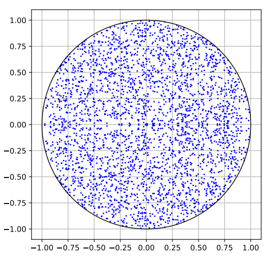
- Sec 5.1 presents a literature review on the field of random polynomials, which has not seen much usage outside of theoretical inquiries
- Sec 5.2 presents the architecture of the IIRNet MLP
- Sec 5.3 mentions the baselines against which IIRNet's accuracy will be measured
- Sec 5.4 outlines the three experiments to be run that will measure how the performance of IIRNet is affected by the filter families used to train it, the size of the MLP, and the order of filters used to train it
- Sec 5.5 outlines three sets of filter families used to evaluate IIRNet
- Sec 5.6 discusses the results of these experiments and their impact

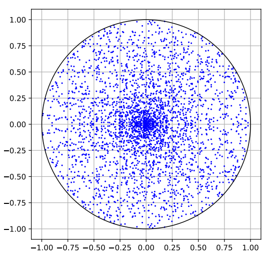
5.1 Random Polynomials

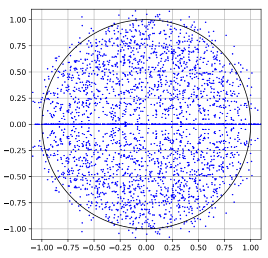
Training neural network filter estimators relies on the generation of a dataset of random digital IIR filters. While sampling random filters for this training process may appear straightforward, it was found that the sampling method plays an important role in generalization. Implicit in the process of random filter generation is a random sampling of polynomials, as demonstrated in Equation (2.9). Figure 5.2 shows the root placements of these polynomials sampling methods for 100 filters with degree 32.


(a) Normal Coefficients


(b) Normal Biquads


(c) Uniform Disk


(d) Uniform Magnitude


(e) Characteristic Polynomial

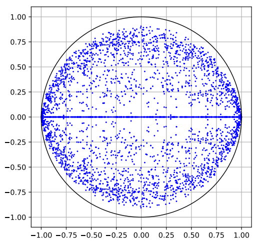

(f) Uniform Parametric EQ

Figure 5.2: Root placements of 100 randomly sampled 32nd degree polynomials.

A. Polynomials with normal coefficients — Given a degree N polynomial $c_N x^N + \cdots + c_1 x + c_0$, sample each c_i from the normal distribution $\mathcal{N}(0, 1)$. For sufficiently large N, the roots of this polynomial converge to the unit circle Hammersley [1956]. Most roots are approximately $\frac{1}{N}$ away from the unit circle Shepp and Vanderbei [1995], and some roots are approximately $\frac{1}{N^2}$ away from the unit circle Michelen and Sahasrabudhe [2020]. Roughly $\frac{2}{\pi} \log N$ roots fall on the real line Kac [1943], most of which are close to 1 and -1. The closest real root to the unit circle is approximately $\frac{1}{N}$ away Michelen [2021]. Much of the behaviour is unchanged if the coefficients are other distributions Ibragimov and Maslova [1971], Bharucha-Reid and Sambandham [2014], Tao and Vu [2015b]. Under very general conditions, the zeros of these polynomials experience repulsion Tao and Vu [2015b]. As long as the polynomial and its derivative are not both likely to be small at the same time, the roots repel each other Tao and Vu [2015b].

B. Biquads with normal coefficients — Given a desired polynomial order N, sample $\frac{N}{2}$ second order polynomials $b_{2,i}x^2 + b_{1,i}x + b_{0,i}$, with $b_{j,i}$ independently sampled from $\mathcal{N}(0, 1)$ and multiply them together. This is a process where roots are sampled independently in pairs, which means the roots of the derivative polynomial are uniformly independent in the same way Pemantle and Rivin [2013], Kabluchko [2015]. A Monte Carlo simulation with 10^8 iterations suggests that about 64.8% of roots sampled using this method are real.

C. Polynomials with uniformly sampled roots in the unit disk — Given a desired order N, sample $\frac{N}{2}$ roots in the complex plane using the following procedure: take θ uniform in $[0, 2\pi]$ and $r = \sqrt{U}$ where $U \sim \text{Unif}[0, 1]$. Then, select these roots' complex conjugates as the remaining $\frac{N}{2}$ roots. Similar to (B), the roots of the derivative polynomial are uniformly independent in the same way Pemantle and Rivin [2013], Kabluchko [2015], with no expected density of real roots given this sampling.

D. Polynomials with roots sampled uniformly in magnitude and argument — Given a desired polynomial order N, sample $\frac{N}{2}$ roots in the complex plane using the following procedure: take θ uniform in $[0, 2\pi]$ and take r uniform in $[0, 1]$. Then, select these roots' complex conjugates as the remaining $\frac{N}{2}$ roots. Similar to (C), the roots of the derivative polynomial are uniformly independent in the same way Pemantle and Rivin [2013], Kabluchko [2015]. There is no expected density of real roots given this sampling. Compared to the roots sampled in (C), these roots will exhibit a greater density closer to the origin than the unit circle.

E. Characteristic polynomial of a random matrix — Given a desired polynomial order N, take a random matrix $A \in \mathbb{R}^{N \times N}$ whose entries are sampled i.i.d. from $\mathcal{N}(0, 1)$ and use its eigenvalues (rescaled by $\frac{1}{N}$) as the roots of the desired polynomial Ginibre [1965], Mehta [1967]. These roots exhibit a repulsion from one another Tao and Vu [2015b]. The eigenvalues converge to the unit disk for various distributions of entries Tao and Vu [2010]. The characteristic polynomial of this random matrix has roughly $\sqrt{2N/\pi}$ real roots Edelman et al. [1994]. It is known this behaviour persists for a family of random variables whose first four moments match those of the Gaussian (i.e. $E[X] = 0, E[X^2] = 1, E[X^3] = 0, E[X^4] = 3$) Tao and Vu [2015a], but still open in cases such as if the coefficients are $\{-1, +1\}$ with equal probability Vu [2020].

F. Uniform parametric EQ — Given a desired polynomial order N, uniformly sample the parameters of a parametric EQ made up of one low shelf section, one high shelf section, and $\frac{N-4}{2}$ peaking filters Nercessian [2020]. The uniformly sampled parameters include each section's corner/center frequency, gain, and Q factor.

5.2 IIRNet Architecture Design

The goal is to train a neural network to learn a mapping f_θ that takes a desired magnitude response $X \in \mathbb{R}^F$ sampled at F linearly spaced frequencies over $[0, \frac{f_s}{2}]$, where f_s is the system sample rate, and estimates an N^{th} order digital IIR filter with a magnitude response $\hat{X} \in \mathbb{R}^F$. For simplicity, N is fixed to be even. This cascade of biquads can then be represented by a scalar gain $G \in \mathbb{R}$, and a set of K second-order sections comprised of K complex poles $P = \{p_0, ..., p_{K-1} \mid p_k \in \mathbb{C}\}$ and K complex zeros $Z = \{z_0, ..., z_{K-1} \mid z_k \in \mathbb{C}\}$, where $K = N/2$. Thus the network learns a mapping $f_\theta(X) \to G, P, Z \to \hat{X}$, with $X, \hat{X} \in \mathbb{R}^F$ as shown in Figure 5.1. Each pole and zero is paired with its complex conjugate to ensure each biquad has real-valued coefficients. Thus the kth biquad takes the form

$$H_k(z) = \frac{1 - 2Re(z_k)z^{-1} + |z_k|^2 z^{-2}}{1 - 2Re(p_k)z^{-1} + |p_k|^2 z^{-2}}. \tag{5.1}$$

Estimating a system gain G rather than an individual gain for each second order section reduces the total number of parameters without loss of generality, and was found to aid stability in training higher order models. Additionally, forcing the system gain $G \in (0, 100)$ aids training stability by applying the sigmoid function to IIRNet gain estimate and then multiplying by 100. To ensure a minimum phase filter, the estimated poles p_k and zeros z_k are rescaled according to Nercessian et al. [2021] as shown in (5.2). To further stabilize training, a constant $\epsilon = 10^{-8}$

was added to prevent root placement at the origin or on the unit circle.

$$
\begin{aligned}
p_k &\leftarrow \frac{(1-\epsilon)\cdot p_k \cdot \tanh(\ |p_k|\ )}{|\ p_k + \epsilon\ |} \\
z_k &\leftarrow \frac{(1-\epsilon)\cdot z_k \cdot \tanh(\ |z_k|\ )}{|\ z_k + \epsilon\ |}
\end{aligned}
\tag{5.2}
$$

During training, the network is tasked with minimizing a loss function $\mathcal{L}$ that measures the distance between the input and estimated magnitude response. The mean squared error of the log of the magnitude responses of the estimated and target magnitude responses over a set of F linearly spaced frequencies $[0, \frac{f_s}{2}]$ is used.

$$
\mathcal{L} = \frac{1}{F}\Big|\Big|\log(\hat{X}) - \log(X)\Big|\Big|_2^2 \tag{5.3}
$$

The complex response of each second order section was calculated by performing the DFT on the numerator and denominator polynomials with zero padding, and dividing the result. This allows for parallelized computation, as opposed to the sample-based gradient optimization in previous works [Kuznetsov et al., 2020, Pepe et al., 2020]. The base IIRNet architecture is composed of 2 linear layers with hidden dimension D, each followed by layer normalization [Ba et al., 2016] and LReLU with $\alpha = 0.2$. The final layer has no activation, and projects the hidden dimension to the number of filter parameters, which is a function of the filter order. The estimation of complex values is treated as the individual estimation of their real and imaginary components.

5.3 Baselines

Two baselines were considered to benchmark IIRNet against existing methods: the modified Yule-Walker [Chan and Langford, 1982] method and a stochastic gradient descent (SGD) method. For the SGD approach the same biquad parameterization and loss function as IIRNet were used, but instead randomly initialized a vector with G, P, Z parameters that are optimized over a number of gradient steps using a learning rate of $5 \cdot 10^{-4}$. The number of gradient steps were then varied to observe the impact on run-time as well as accuracy.

5.4 Experiments

AdamW [Kingma and Ba, 2015, Loshchilov and Hutter, 2017] was used for SGD and IIRNet was trained for 500 epochs with a batch size of 128, where 1 epoch is defined as $20,000$ random filters, equating to a total of 10 million filters. The target magnitude response was evaluated over $F = 512$ linearly spaced frequencies. To aid stability, all responses are clipped $X \in [-128\,\text{dB}, 128\,\text{dB}]$ and then scaled between $[-1, 1]$. All models were trained with an initial learning rate of 10^{-5} unless otherwise noted, and the learning rate decayed by a factor of 1/10 at 80 and 95% through training. Gradient clipping is applied where the norm of the gradients exceeded 0.9. Three experiments were conducted to investigate the behaviour of IIRNet, training a total of 19 models. Code for these experiments along with pre-trained models is provided.[1]

Filter family — To investigate the impact of the random filter sampling method seven models were trained, training each on a different family of random 16th order filters (A-F) as described in Section 5.1, with the final model trained using all of the families together (G). For these experiments, each linear layer had $D = 1024$ hidden units, and each model was trained to estimate a 16th order biquad cascade.

Model size — The size of the linear layers within IIRNet has a direct impact on the inference time, which is of interest for online and real-time applications. The impact of the model size on the run-time and accuracy was investigated by training another seven models using hidden sizes $D \in 64, 128, ..., 4096$. These models were trained an equal number of random filters from all of the families (G). All timings were performed on CPU and averaged over a total of 1000 runs, using a machine with an AMD Ryzen Threadripper 2920X.

Filter order — IIRNet predicts a fixed order filter given a desired magnitude response, which means that a different model must be trained for different filter order estimations. To investigate the performance of IIRNet as a function of the filter order, another five models were trained, varying both the order of the random filters used in training, and the filter order estimated by IIRNet. These models used $D = 2048$ hidden units in each linear layer and were trained again with random filters from all families (G). Since the training of models that estimate higher order filters ($N \geq 32$) was found to be more unstable, all such models were trained with an initial learning rate of 10^{-6}.

Training Method	**Random polynomial families**							**Real-world**		**Avg**
	A	B	C	D	E	F	G	HRTF	Gtr. Cab.	
Modified Yule-Walker ($N = 16$)	12.84	32.46	16.67	124.23	6.80	1.40	19.73	**1.19**	60.86	30.69
A. Normal coefficients	**4.38**	6.80	6.22	23.11	1.42	1.11	5.07	1.35	6.73	6.24
B. Normal biquads	13.19	**2.70**	0.21	1.29	2.14	0.57	2.64	2.40	6.86	3.55
C. Uniform disk	193.81	328.79	**0.08**	1.19	8.91	50.42	83.32	263.06	1203.40	237.00
D. Uniform magnitude disk	175.81	279.54	0.09	**0.54**	11.25	61.41	76.37	250.38	1111.05	218.49
E. Characteristic polynomial	22.95	32.66	0.35	2.44	**0.81**	0.72	6.81	11.02	138.99	24.08
F. Uniform parametric EQ	19.33	12.84	3.06	17.84	3.52	**0.21**	6.89	3.79	17.50	9.44
G. All families	6.24	2.89	0.11	0.67	1.12	0.34	**1.28**	1.40	**5.59**	**2.18**

Table 5.1: Average dB MSE for IIRNet trained using different families of random 16th order ($K = 8$) filters.

5.5 Evaluation Datasets

Three different sets of filters were used to evaluate the models. First, 1000 random filters from each of the 7 proposed random filter families (A-G) were evaluated. Then, models were measured on how they generalized to distributions of filters not seen during training, as well as matching the magnitude response of real-world filters such as measured head-related transfer functions (HRTFs) and guitar amplifier cabinets. Though phase is an integral part of the HRTF, some studies suggest that the HRTF can be reproduced within perceptual tolerance under certain conditions via a minimum phase magnitude response plus delay match [Kulkarni et al., 1999]. Guitar cabinets combine loudspeakers with guitar amplification circuits for use in creative settings within music production. The impulse response of these cabinets can

[1] `https://github.com/csteinmetz1/IIRNet`

Method	**Params.** Million	**Time** ms	**(G)** dB MSE	**HRTF** dB MSE	**Gtr. Cab.** dB MSE
MYW	-	9.00	19.73	1.19	60.86
SGD (1)	-	7.75	2458.28	3165.43	5648.83
SGD (10)	-	58.21	998.20	1393.29	2362.49
SGD (100)	-	578.52	11.74	3.49	5.67
SGD (1000)	-	5784.94	9.49	0.76	2.25
IIRNet 64	0.04	0.28	3.70	2.74	7.22
IIRNet 128	0.09	0.29	2.95	2.41	7.11
IIRNet 256	0.21	0.30	2.08	2.03	6.29
IIRNet 512	0.55	0.36	1.51	1.69	6.54
IIRNet 1024	1.63	0.71	1.29	1.39	5.54
IIRNet 2048	5.35	1.87	1.16	1.52	5.02
IIRNet 4096	19.1	4.65	1.11	1.38	5.86

Table 5.2: Comparison of average dB MSE and runtime in milliseconds fitting 16th order filters with other IIR filter design methods.

Train	**Test order (G)**					**HRTF**	**Gtr. Cab.**
Order	4	8	16	32	64		
4	1.21	7.65	20.30	75.28	196.19	11.77	19.20
8	0.37	1.59	6.20	24.98	80.10	6.08	11.96
16	0.22	0.68	2.13	9.55	34.76	1.97	6.12
32	**0.17**	**0.39**	**0.98**	**4.82**	**21.32**	**0.66**	**1.92**
64	1.96	2.07	2.69	7.49	22.61	3.29	4.70

Table 5.3: dB MSE evaluating IIRNet with different estimation orders.

then be used for digital emulation of the linear behaviour of these devices. 187 HRTFs were sourced from the IRCAM-Listen HRTF Dataset[2] and 32 guitar cabinet impulse resposnes were sourced from Kalthallen Cabs[3]. All impulse responses were resampled to 16-bit 44.1kHz and a Savitzky-Golay filter [Luo et al., 2005] was used to smooth the magnitude responses before input to IIRNet.

5.6 Results and Discussion

The MSE between the estimated and target response are reported using a dB scale, enabling a more interpretable analysis of the error. Experiments with different random filter families in Table 5.1 show that training on a specific family of random filters resulted in the best performance when evaluating on that filter family. Furthermore, it was found that training on certain families (A, B, F) rather than others (C, D, E) resulted in better performance on real-world filters. This supports the claim that the method for constructing random filters is a significant consideration in training this type of model. Notably, IIRNet trained on all filter families (G) achieved the lowest combined MSE across all datasets, indicating that training on multiple families is superior to training on any single family alone.

The performance of these models is also compared against the MYW approach, as shown in the first row of Table 5.1. Here MYW is used to fit the desired response specifying the filter

[2]`http://recherche.ircam.fr/equipes/salles/listen/`
[3]`https://cabs.kalthallen.de`

order $N = 16$, the same as the target filter. This approach performs worse than IIRNet trained with (G) across all of the random polynomial families, along with the guitar cabinet responses. However, it was found that MYW outperforms other methods on the HRTF dataset. These results point to MYW performing better when the overall range of the magnitude response is more limited, but this approach may struggle when the response has a much larger range in the magnitude space.

The run-time and accuracy of variants of IIRNet are compared to an SGD and MYW approximation on identical datasets in Table 5.2. Both the run-time and accuracy increase as we increase the size of IIRNet, as expected. All versions of IIRNet are both faster and more accurate across the set containing all random filter families (G) as compared to both SGD and MYW. On the real-world filter estimation tasks, SGD with 1000 iterations outperforms MYW and even the largest IIRNet model, but has a run time orders of magnitude higher. MYW beats all other approaches on the HRTF estimation task, but performs worse than even the smallest IIRNet model across all random filters families and guitar cabinet estimation.

Since IIRNet is trained to estimate filters of a fixed order, it was evaluated how performance changed as a function of the estimation order. Table 5.3 demonstrates that in general, increasing the estimated filter order of IIRNet improves estimation accuracy at all orders less than or equal to the training order. However, it was found training models that estimate filters with order $N \geq 64$ challenging, often leading to instability. As a result, the model trained to estimate 64^{th} order filters diverged, and hence performs worse than the 32^{nd} order model.

While these results demonstrate that IIRNet produces accurate estimates of both unseen random and real-world filters, this approach has some limitations including fixed order filter estimates, consideration only of the magnitude response, and the inability to apply additional design constraints. Future work could investigate a formulation of the loss function that also considers phase, along with architectural adjustments that may support variable filter order estimation. However, it may be possible to address some of these limitations by using IIRNet simply as a method for generating an initial estimate that can be refined with more flexible iterative techniques.

Chapter 6

Memoryless Distortion Effect Matching

This section outlines procedures for matching memoryless distortion audio effects given an input signal $x[n]$ and output $y[n]$. As mentioned in Sec 3.5, there is a wealth of approaches to modelling audio distortion effects that range in complexity and interpretability.

Differentiable whitebox methods like those presented in Parker et al. [2019] can emulate the differential equations that govern analogue distortion circuits on a sample-by-sample basis. The authors' novel feedforward ANN formulation enables avoids the often costly and difficult to train autoregressive networks such as RNNs. To do this, the authors collect not only input/output samples from the distortion effects, but also internal measurements of the circuit's behaviours. The authors baseline their approach on three distortion circuits: a first-order diode clipper, a second-order diode clipper, and a Sallen-Key filter.

On the other hand, differentiable blackbox methods like those presented in Martínez Ramírez [2021] require no measurements of analogue circuitry and learn their behaviour strictly from input/output samples. Among the many audio effects modelled, two analogue distortion effects are modelled: the Universal Audio vacuum-tube preamplifier 610-B and the Universal Audio transistor-based limiter amplifier 1176LN. While this approach reduces the complexity of the dataset needed to emulate a distortion effect, the complexity of the algorithm itself must increase to compensate. The best performing architectures in this task employed neural networks with memory, thus drastically increasing their training time and complexity compared to the MLPs of Parker et al. [2019].

A lightweight approach that splits the difference between the interpretability of Parker et al. [2019] with the ease of data collection in Martínez Ramírez et al. [2021] is the greybox, DDSP approach in Nercessian et al. [2021]. As described in Sec 3.5.4, this approach uses a cascade of differentiable parametric EQs and tanh nonlinearities to model distortion effects. The authors choose emulate a Boss MT-2 distortion pedal using 10 cascaded filtering stages.

This greybox approach allows for an interpretable model of a distortion effect, as the parameters of the parametric EQs and tanh nonlinearities are readily readable. However, this model would be difficult to tweak as it would involve a user to modify the parameters of six parametric EQs and saturating nonlinearities.

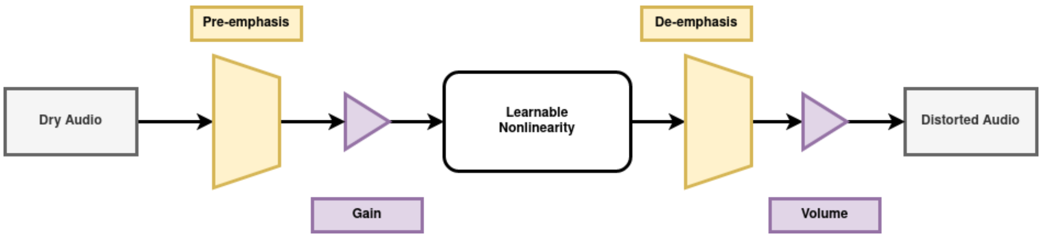


Figure 6.1: Block diagram of the proposed DDSP W-H distortion effect.

This chapter presents a novel differentiable Wiener-Hammerstien (W-H) DDSP memoryless distortion effect. This model is smaller and more interpretable than the methods cited above, with the addition of being easily modified by a user. In Sec 6.1, the W-H model is defined. Sec 6.2 presents a dataset of audio samples for three different software distortion effects. This dataset will be used to evaluate the modelling power of the differentiable W-H model. Then, Sec 6.3 details the formulation of each processing block in the DDSP W-H model. Here, several different families of learnable nonlinear waveshaping functions will be discussed. These include two novel formulations: the SumTanh family and PowTanh family. Sec 6.4 describes the optimization procedure used to fit an input/output pair of audio samples, as well as notes on initialisation. Sec 6.5 presents the results of an objective evaluation and a perceptual evaluation of the algorithm, and Sec 6.6 discusses the results.

6.1 Wiener-Hammerstein Models

The typical W-H model consists of a linear block, a nonlinear block, and a linear block cascaded in series. The W-H models in this work are time-invariant, formulated using graphic equalizer pre-emphasis and de-emphasis filters for the linear blocks and a parameterized waveshaping function as the nonlinearity. While previous literature focused on power series and Chebyshev polynomials to model the nonlinear blocks [Novak et al., 2010b,a], in this work several other waveshaping functions are investigated. Figure 6.1 shows a diagram of the proposed W-H model used in this work.

6.2 Evaluation Dataset

A novel dataset of processed electric guitar samples following the same procedure described in Comunità et al. [2021] was assembled, using unprocessed recordings from the IDMT-SMT-Audio-Effects dataset [Stein et al., 2010].

The source dataset[1] includes monophonic (624 single notes) and polyphonic (420 intervals and chords) recordings (wav - 44.1kHz, 16bit, mono) from 2 different electric guitars, each with two pick-up settings and up to 3 plucking styles. The monophonic recordings cover the common pitch range of a 6-string electric guitar, and the polyphonic samples were obtained mixing single notes recordings to generate 2-note intervals and 3- or 4-note chords. All samples

[1]https://www.idmt.fraunhofer.de/en/business_units /m2d/smt/audio_effects.html

Table 6.1: Plugins used in this work

Designer	*Plugin*	*Emulation of*	*Id*
Audified	Multidrive Pedal Pro	ProCo Rat	RAT
Mercuriall	Greed Smasher	Mesa/Boogie Grid Slammer	MGS
Analog Obsession	Zupaa	Vox Tone Bender	VTB

Table 6.2: Plugin settings used to generate the dataset

Id	*Level*	*Gain*	*Tone/Eq*
RAT	[1.0]	[0.2, 0.5, 1.0]	[0.2, 0.8]
MGS	[1.0]	[0.2, 0.5, 1.0]	[0.2, 0.8]
VTB	[1.0]	[0.1, 0.2, 0.5, 0.8, 1.0]	—

are 2 seconds long. The monophonic recordings required removal of background noise before the note onset, which was obtained using a python script together with *Librosa*'s onset detection function [McFee et al., 2015].

To assemble the dataset an overdrive, distortion and fuzz plug-ins (see Table 6.1) designed to emulate some of the most iconic and widely used analogue guitar effect pedals were selected. By selecting 3 different types of distortion plug-ins from 3 different developers, it was desired to cover a wide range of timbres and designs while keeping the amount of data limited. All the plugins have 2 or 3 controls and, regardless of the specific name adopted by the designer, the controls can be identified by their processing function: Level, Gain, Tone/Equalisation. A summary of the controls and settings is shown in Table 6.2. These values were chosen as they were found to be perceptually distinct from one another. The samples' were processed in MATLAB - making use of its VST plugin host features - and both unprocessed inputs and processed outputs were normalised to 0dBFS.

6.3 Methods

6.3.1 Learnable Graphic EQ

Similar to the formulation in Colonel and Reiss [2021], separate 20-band finite impulse response (FIR) graphic EQs are learned for the pre-emphasis and de-emphasis linear blocks [Välimäki and Reiss, 2016]. These EQs are calculated using the frequency sampling method as in Engel et al. [2019]. First, a frequency transfer curve is specified. The inverse short-time Fourier transform (ISTFT) of this magnitude response is taken using a zero-phase response to obtain the filter's impulse response (IR). Afterward the EQ is applied by multiplying the magnitude response of this new IR with the windowed STFT of the input audio signal.

The 20-band graphic EQ can be characterized using a 20 dimensional $\Theta_{EQ\ gains}$. The 20 values specify the gain of each octave band filters, which are centered at 40, 65, 80, 130, 200, 270, 400, 540, 800, 1000, 1500, 2000, 3000, 4000, 6000, 8000, 12000, and 16000 Hz respectively. Shelving filters are used for frequencies below 40Hz and above 16000 Hz that match the attenuation specified at the lowest and highest octave band respectively.

These 20 values are transformed via

$$\Theta_{EQ\ gains} \leftarrow \sigma(\Theta_{EQ\ gains}) \tag{6.1}$$

where σ denotes the sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{6.2}$$

The values in the transformed $\Theta_{EQ\ gains}$ range from $(0, 1)$ due to the bounds of the sigmoid function.

Finally a piecewise linear frequency transfer curve Θ_{EQ} is constructed using linear interpolation between the octave band attenuations specified by $\Theta_{EQ\ gains}$. Thus the EQ module's frequency transfer curve is bounded from (0,1) at all points. The estimated values are initialized with random uniform noise from [-1,1], which initializes the octave band gains from -6dB to -1dB.

Because this EQ formulation only allows for the attenuation of frequencies, a gain value is specified in tandem with the pre-emphasis filter, and a volume value is specified in tandem with the de-emphasis filter. It was found that this decoupling of gain and attenuation helps stabilize the optimization.

6.3.2 Waveshaping nonlinearity functions

Tanh Nonlinearity

A hyperbolic tangent (Tanh) with DC offset is used as the baseline nonlinearity in this work. The tanh function

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{6.3}$$

is often used to model distortion effects due to its saturating behavior towards $\pm\infty$. To enable the modelling of nonsymmetric distortion, a DC offset b_{DC} can be applied before the tanh nonlinearity. However, this offset must be removed after the nonlinearity to ensure the output signal maintains no DC offset

$$f(x, b_{DC}) = \tanh(x + b_{DC}) - \tanh(b_{DC}) \tag{6.4}$$

SumTanh Nonlinearity

Proposed in this work is a family of functions called a "harmonic sum of tanh functions" with DC offset (SumTanh)

$$f(x) = a_1 \tanh(x) + a_2 \tanh(2x) + \cdots + a_{n-1} \tanh((n-1)x) + a_n \tanh(nx) \tag{6.5}$$

As a weighted sum of tanh functions, the SumTanh family exhibits saturation towards $\pm\infty$.

As a sum of odd functions, the SumTanh family are odd functions, meaning they can model symmetric distortions. Note that $a_c \tanh(c \times 0) = 0$, meaning the waveshaper introduces no DC offset. To model asymmetric distortions, a DC offset can be introduced via

$$f(x, b_{DC}) = a_0 + a_1 \tanh(x + b_{DC}) + \cdots + a_{n-1} \tanh((n-1)(x + b_{DC})) + a_n \tanh(n(x + b_{DC})) \tag{6.6}$$

with $a_0 = -\sum_{c=1}^{n} a_c \tanh(c \times b_{DC})$ to remove the DC component after the nonlinearity. For stability, a_1 is initialized close to 1 and a_c are initialized close to 0 otherwise. During optimization, $f(x)$ is normalized such that $\max(|f(x)|) = 1$

PowTanh Nonlinearity

Also proposed in this work is a family of functions called a "power sum of tanh functions" (PowTanh)

$$f(x) = a_1 \tanh(x) + a_2 \tanh(x^2) + \ldots a_{n-1} \tanh(x^{n-1}) + a_n \tanh(x^n) \tag{6.7}$$

As a weighted sum of tanh functions, the PowTanh family exhibits saturation towards $\pm\infty$. However, as a sum of even and odd functions the PowTanh can model both symmetric and asymmetric distortions. Note that $a_c \tanh(0^c) = 0$, meaning the waveshaper introduces no DC offset. As such, no DC offset is included in this parameterization. For stability, a_1 is initialized close to 1 and a_c are initialized close to 0 otherwise. During optimization, $f(x)$ is normalized such that $\max(|f(x)|) = 1$

Fourier Series

Because they are well known for their modelling capability, a parameterized Fourier series waveshaper is investigated. An N^{th} degree Fourier waveshaper takes the form

$$\begin{aligned} f(x) = a_1 \sin(x) + a_2 \sin(2x) + \cdots + a_{\frac{n}{2}} \sin(\frac{n}{2}x) + \\ b_0 + b_1 \cos x + b_2 \cos(2x) + \cdots + b_{\frac{n}{2}} \cos(\frac{n}{2}x) \end{aligned} \tag{6.8}$$

As a sum of even and odd functions, this waveshaper can model both symmetric and asymmetric distortions. However, this waveshaper can introduce a DC offset due to its cosine components, and thus b_0 is fixed to $-\sum_{c=1}^{n/2} b_c$.

Traditionally, the support of a Fourier series is $[-\pi, \pi]$. However, a fourier series can exhibit overshoot behavior towards the ends of that support. Therefore, the input audio signal to the waveshaper is normalized between $[-0.9\pi, 0.9\pi]$ to avoid overshoot artefacts. In practice this means the gain parameter in Figure 6.1 is ignored. As the waveshaper is expected to model distortion, the coefficients of the Fourier series are initialized with an N^{th} order approximation to a square wave. During optimization, $f(x)$ is normalized such that $\max(|f(x)|) = 1$

Legendre Polynomials

Because they are well known for their modelling capability, a parameterized Legendre polynomial waveshaper is also investigated. Legendre polynomials are a family of polynomials $P_n(x)$

orthogonal on $[-1,1]$ with $P_n(1) = 1$. From Rodrigues' formula these polynomials can be expressed as

$$P_n(x) = \sum_{k=0}^{n} x^k \binom{n}{k} \binom{\frac{n+k-1}{2}}{n} \tag{6.9}$$

and the waveshaper takes the form

$$f(x) = a_0 + a_1 P_1(x) + a_2 P_2(x) + \cdots + a_n P_n(x) \tag{6.10}$$

As a sum of even and odd functions, this waveshaper can model both symmetric and asymmetric distortions. However, this waveshaper can introduce a DC offset due to its constant components, and thus a_0 is fixed to $-\sum_{c=1}^{n} a_c P_c(0)$.

Similar to the Fourier series, Legendre polynomials can exhibit overshoot behavior towards the edges of its support. Therefore, the input audio signal to the waveshaper is normalized between $[-0.9, 0.9]$ to avoid overshoot artefacts. In practice this means the gain parameter in Figure 6.1 is ignored. As the waveshaper is expected to model distortion, the coefficients of the Legendre polynomials are initialized with an N^{th} order approximation to the square wave. During optimization, $f(x)$ is normalized such that $\max(|f(x)|) = 1$

6.4 Optimization

Stochastic gradient descent (SGD) is used to update the W-H parameters to fit a dry/wet audio pair. These parameter include the attenuations of each band in both pre-emphasis and de-emphasis filters, the gain parameter, each coefficient in the waveshaper, and the volume parameter. The cost is calculated by passing a dry audio sample through the estimated W-H model and measuring the distance between the estimated and target audio signal. SGD is performed using the Adam method with an initial learning rate of 10^{-3}. The cost function chosen is multiscale spectrogram loss with window sizes 46ms, 12ms, and 3ms [Engel et al., 2019]. The optimizations are allowed to run for a maximum of 40000 iterations. Early stopping is employed with a patience of 1000 iterations. The learning rate is dropped to 10^{-4} when the first early stopping is reached, or when the optimization reaches 20000 iterations.

6.5 Results

6.5.1 Objective Evaluation

The dataset outlined in Section 6.2 was used to measure the objective performance of each waveshaping method. Each waveshaper was modeled using 10 degrees of freedom. Presented in Table 6.3 are the average multiscale spectrogram losses measured on each waveshaping method across each plugin and across all plugins. Across all plugins, the PowTanh method performs best. Similarly, the PowTanh method outperforms all other methods on the RAT and VTB plugins. On the MGS plugin, the SumTanh performs best. Both PowTanh and SumTanh methods outperform the Tanh baseline on all tasks. Notably, the Fourier and Legendre waveshapers did not outperform the Tanh baseline on any of the tasks.

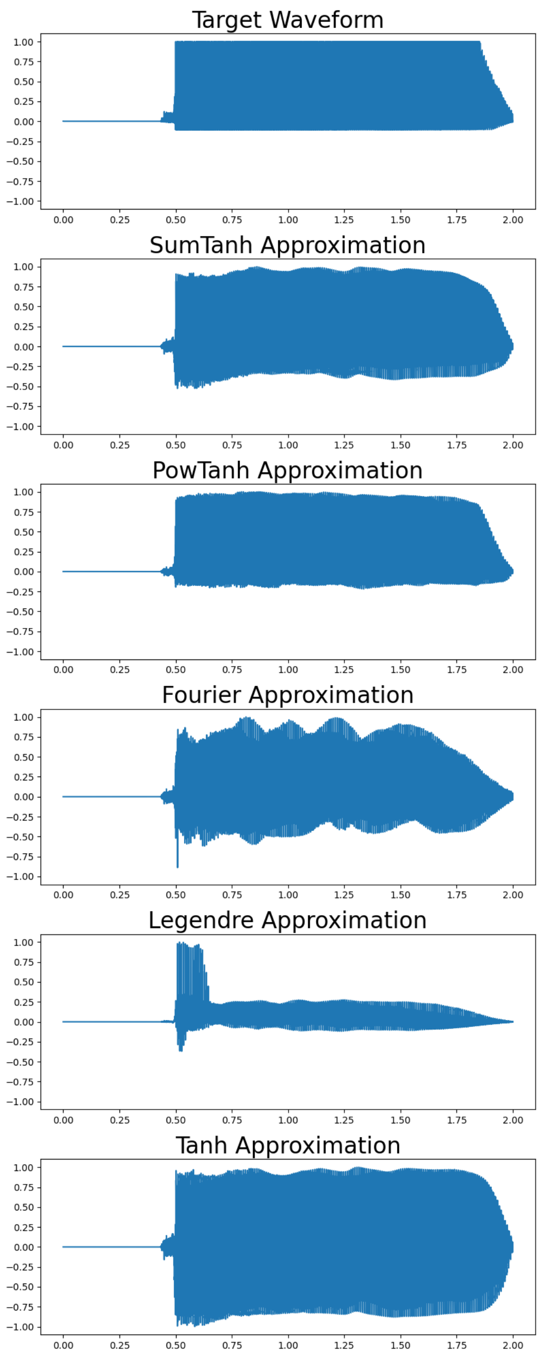


Figure 6.2: Example of estimated waveforms for each waveshaping model on a VTB example

Table 6.3: Mean multiscale spectrogram loss of the waveshapers evaluated across pedals

Waveshaper	*RAT*	*MGS*	*VTB*	*Total*
Powtanh	1.321	**0.559**	**2.117**	**1.332**
Sumtanh	**1.321**	0.573	2.400	1.431
Fourier	1.588	0.603	2.686	1.625
Legendre	1.703	0.640	2.893	1.746
Tanh	1.353	0.597	2.478	1.476

Table 6.4: Results of pairwise comparison of waveshaping method architecture on perceptual similarity rating across all stimuli, with Bonferroni Correction, o > 0.9, *<0.001 ·=no comparison. No waveshaping method was found to be perceptually indistinguishable from the reference.

	Ref	SumTanh	PowTanh	Fourier	Legendre	Tanh
Ref	·	*	*	*	*	*
SumTanh	*	·	o	*	*	o
PowTanh	*	o	·	*	*	o
Fourier	*	*	*	·	*	*
Legendre	*	*	*	*	·	*
Tanh	*	o	o	*	*	·

6.5.2 Perceptual Evaluation

A listening test was conducted using webMUSHRA on a subset of 12 samples from the dataset [Schoeffler et al., 2018]. This subset consists of two unique monophonic and two unique polyphonic stimuli passed through each of the three distortion plugins. No plugin parameter settings were repeated across any of the stimuli. These stimuli were chosen to cover a broad subset of the dataset and to avoid biasing results towards a specific effect, parameter setting, or input signal.

Participants of the study were told:

> You will be provided with a reference at the top of the page. The task is then to rate the stimuli below based on their quality compared to the reference. Adjust the sliders for each example to rate the quality, and use the whole scale when possible. A perfect score should constitute a signal that exactly matches the reference.

Participants were asked to rate how closely each of the learned W-H models' outputs matched a reference signal, with 0 representing a poor match and 100 representing a perfect match. The reference signal was included among the stimuli. Thus for each of the 12 reference stimuli, the participants were presented with samples reverse engineered using the SumTanh, PowTanh, Tanh, Fourier, and Legendre waveshaping families as well as the hidden reference.

A total of 17 participants took part in the study, with an average age of 31 years and standard deviation of 5.34. 8 participants identified as men, 7 as women, and 3 as nonbinary or gender nonconforming. 11 participants reported having at least 5 years of experience with music production or audio engineering, 2 reported having 3 years experience, and 4 participants reported no experience. No participants reported any diagnosed hearing impairments. Box and whisker plots of the participants' ratings are presented in Figure 6.3, broken down by the waveshaping family used for reverse engineering the stimuli.

The analysis that follows is adapted from the perceptual study presented in Moffat and Reiss [2018]. The null hypothesis is that the perceptual evaluation scores are from the same distribution. A one-way ANOVA, with Bonferroni correction, shows for all stimuli that the

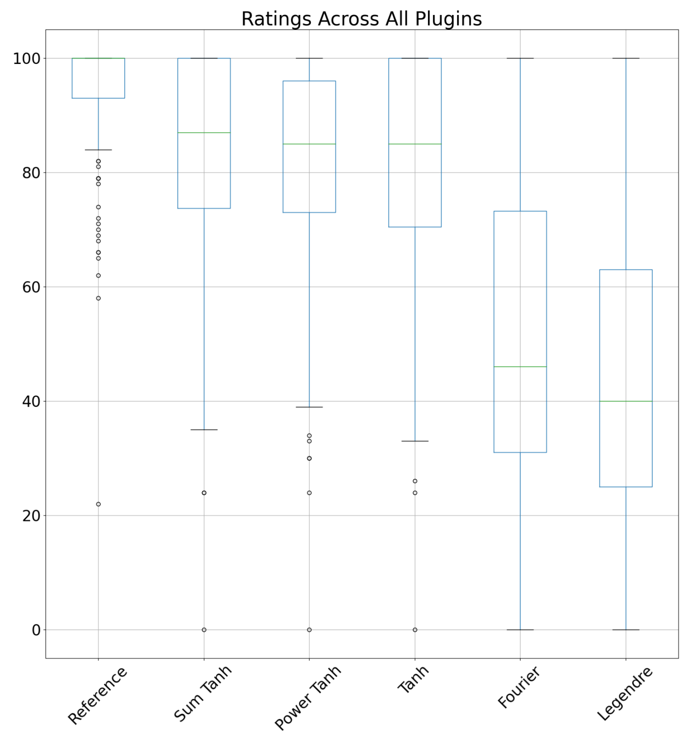


Figure 6.3: Box and whisker plots of the median, standard deviation and 95% confidence intervals of all participant ratings for each waveshaping method across all plugins. No waveshaping method was found to be perceptually indistinguishable from the reference.

effect each waveshaper had on user perception was statistically significant.

With the null hypothesis rejected, a post-hoc Tukey pairwise comparison, with Bonferroni correction to reduce the chance of type I errors, was used. Table 6.4 shows the results of these pairwise comparisons for all architectures used. The pairwise comparisons demonstrate that across all plugins, the perception of each waveshaping method differs significantly from the reference. However, when broken down by plugin type the perception of the SumTanh model does not differ significantly from the reference for the MGS and VTB effects. For the MGS effects the caluclated p-value between the reference stimulus and SumTanh stimulus is 0.372, and for the the VTB effects the calculated p-value is 0.064.

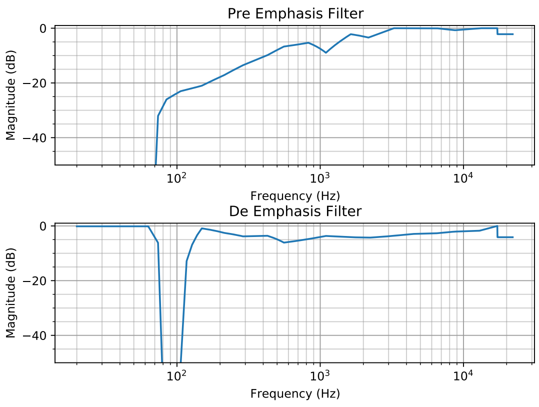
6.6 Discussion

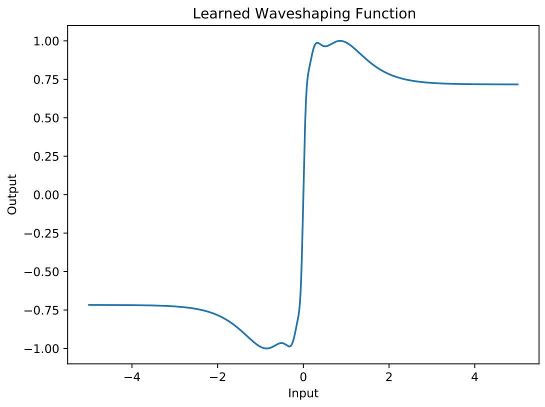
Despite outperforming all models in the objective evaluation, the PowTanh waveshaper did not hold up to perceptual evaluation. Instead, the SumTanh model proved to be the most perceptually accurate model. Both the Fourier and Legendre waveshapers performed worse than the baseline Tanh model in the perceptual evaluations. This demonstrates that MSS loss does not necessarily correlate to perceptual closeness. While Engel et al. [2019] correctly points out that MSS loss can ignore slight discrepancies in phase modelling that a human listener may not notice, others like Turian and Henry [2020] point out that MSS loss can do poorly when matching pitch. Differentiable mesostructural approaches like that shown in Vahidi et al. [2023] or contrastive learning approaches like Manocha et al. [2021] may be more appropriate for this reverse engineering task.

Figure 6.2 shows the estimated waveforms of learned W-H models for each waveshaper given a VTB target. Both the SumTanh and PowTanh models are able to match the asymmetric distortion well. The Fourier model exhibits an obvious tremolo-like artefact, and the Legendre model shows an overshoot behavior during the attack of the sample. These behaviors were typical across the objective study. The Tanh model was unable to learn the asymmetry in the waveform, a behavior that deserves future exploration.

Figure 6.4 shows the learned EQs and waveshaping function for the SumTanh model mentioned above. In this example the learned gain is 16.084, DC offset is -0.026, and volume is 1.170. The pre-emphasis filter appears to mimic a standard high-pass filter with a cutoff frequency of 1kHz, and the de-emphasis filter has a slight attenuation for most of the middle frequencies and a steep notch at 100Hz. The waveshaper learned to have a slight overshoot for values close to 0 and to saturate at a value less than 1. Future work may involve better understanding the trajectories of these learned parameters over the course of the optimization.

Further studies must be undertaken to fully understand where the SumTanh and PowTanh models underperformed in the perceptual evaluation. One potential issue may be that anti-aliasing is not explicitly addressed in either of these models. While the multiscale spectrogram loss would penalize aliasing harmonics, most waveshaping distortion models explicitly account for anti-aliasing. While internal oversampling may be feasible within the DDSP framework, it would certainly be possible to oversample the dry/wet audio pairs and apply a fixed lowpass filter after the de-emphasis filter when fitting parameters. The authors note that computation time to fit an example would scale with the oversampling factor.





(A) (B)

Figure 6.4: Example of learned parameters using the SumTanh waveshaping model on the VTB stimulus. (A) Pre-emphasis filter (top) and de-emphasis filter (bottom), (B) Learned waveshaping function

Chapter 7

Dynamic Range Compression Effect Matching

This section outlines procedures for matching dynamic range compression audio effects given an input signal $x[n]$ and output $y[n]$.

While blackbox neural networks have been used to emulate DRCs [Hawley et al., 2019, Steinmetz and Reiss, 2022], to the author's knowledge only two DDSP DRC has been proposed in the literature [Steinmetz et al., 2022, Wright et al., 2022]. In Steinmetz et al. [2022] the authors implemented a DRC using Pytorch with tunable threshold, ratio, knee width, makeup gain, and a ballistics control. The authors approximate both attack and release time with a joint smoothing parameter that controls a single pole IIR filter to smooth the DRC's attenuation curve. This IIR filter is then approximated using an FIR filter. As stated by the authors, forcing the attack time and release time to be shared restricts the modelling capabilities of the DRC. Research has shown that attack and release settings in a DRC play a role in the perception of sound quality [Wagenaars et al., 1986, Neuman et al., 1998] and style/genre [Bromham et al., 2018].

The work presented in Wright et al. [2022] applies a similar technique to Steinmetz et al. [2022], but adds complexity to the one-pole filter used for smoothing and can separately model attack and release. The authors do this in two ways. In the first, separate attack and release times are used to switch the one pole filter used for smoothing. This approach requires a recursive calculation based on the signal's current and previous values, which slows down training. In the second, a hidden RNN is used to modulate the one pole filter, which also slows training. The authors also model the makeup gain with another RNN, as many analog compressors use a voltage-controlled amplifier for makeup gain application.

As stated in Steinmetz et al. [2022], the difficulty in implementing a DRC using a differentiable framework lies in the recursive nature of the attack and release calculation. This sample-by-sample "differentiation through time" is costly in both time and memory. Thus the methodology presented here seeks to avoid sample-by-sample calculations and instead approximates attack and release passages as smoothing filters applied to a downsampled loudness curve of the audio signal.

The proposed DDSP DRC architecture is detailed in Sec 7.1. This section also includes a

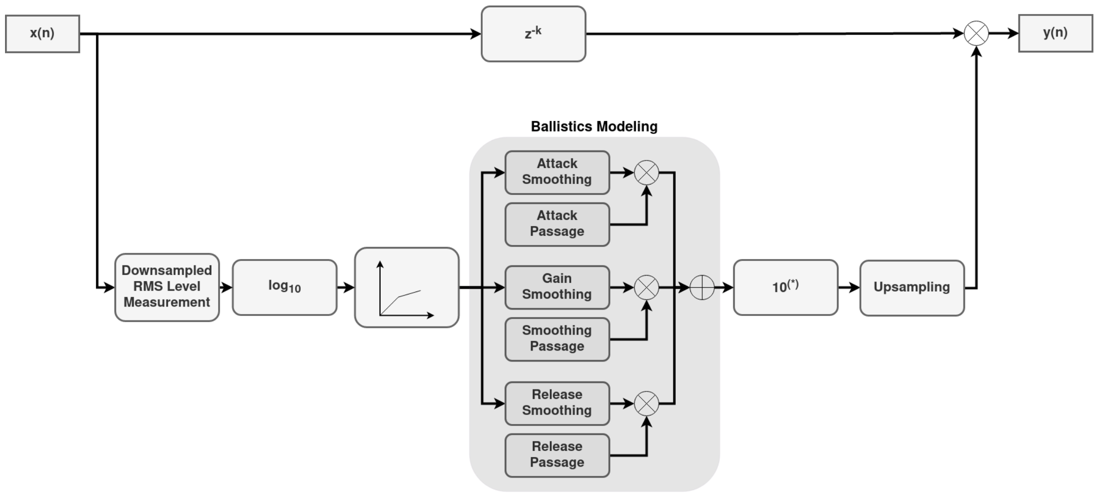


Figure 7.1: Block diagram of proposed DDSP DRC

definition of an approximate moving average filter, which is a novel construct for modelling differentiable attack and release. An experiment to measure the modelling performance of the algorithm is outlined in Sec 7.2. Notes on parameter initialization for the proposed DRC in a reverse engineering task are given in Sec 7.3. Sec 7.4 presents the results of the experiment, and Sec 7.5 discusses said results.

7.1 Proposed DDSP DRC

Refer to Figure 7.1 for a block diagram of the proposed system. Given a fixed length audio signal sampled at 44.1kHz and values for parameters mentioned in Section 2.6 the following steps are used to apply dynamic range compression.

First, a root mean square (RMS) level measurement is calculated using a 5ms window and hop size 0.22ms and converted to dB. This generates a loudness curve measured at 4410 frames per second.

Then, attack and release passages are estimated by finding when this loudness curve crosses the threshold value. Attack passages are calculated by convolving a rectangular window of length τ_{at} with the rising edge of the input signal passing the threshold, and release passages by convolving a rectangular window of length τ_{rt} with the falling edge. The length of these rectangular windows correspond to the attack and release times calculated in frames. These passages finally interfere with one another when they overlap so that the DRC is not simultaneously set to attack and release. Finally, gain smoothing passages are calculated by finding the portions of the loudness curve both above the threshold and outside of attack passages. Thus three masks are produced that are the length of the signal's loudness curve corresponding to attack passages, release passages, and smoothing passages.

Afterwards a compression characteristic is calculated using the threshold, ratio, and knee width for the duration of the signal. This compression characteristic curve is then subtracted from the original signal's loudness curve in order to produce an attenuation curve. Note that this curve measures the dB attenuation per frame that when applied to the original signal produces the characteristic curve.

Given a time constant τ in frames, an approximate moving average filter with support $[0, N]$ takes the form

$$h(x) = \frac{1}{\sum_0^N (\tanh(0.1 * \mathrm{relu}(\tau - x))} \tanh(0.1 * \mathrm{relu}(\tau - x)) \tag{7.1}$$

where $\tanh(x)$ refers to the hyperbolic tangent function and $\mathrm{relu}(x)$ refers to the rectified linear unit. While a multiplicative constant larger than 0.1 would make h(x) more closely approximate a moving average filter, it was experimentally found that the 0.1 scaling factor provides a decent approximation while allowing for gradients to backpropagate through the system.

Three approximate moving average filters are calculated using τ_{at}, τ_{rt}, and τ_{st}, corresponding to the attack action, release action, and gain smoothing action of the DRC. These three filters are convolved in parallel with the attenuation curve, windowed according to the attack/release/smoothing passages mentioned above, and then summed. Afterwards the makeup gain is applied.

Finally the smoothed attenuation curve is converted from dB to a linear scale, upsampled from 4410 frames per second to the original sampling rate using linear interpolation, delayed by 5ms to simulate the lag in level measurement, and applied to the original audio sample via multiplication.

7.2 Experiment

The modeling capability of the proposed method is evaluated using a reverse engineering of dynamic range compression task [Bitzer et al., 2006], in which the parameters of a DRC are inferred using a compressed signal and its dry counterpart. A similar task was proposed in Barchiesi and Reiss [2010], where dynamic effects processing in a multitrack mix was estimated using frame-based polynomial gain estimation. Two signals were chosen to test the performance of the method: a signal proposed in Bitzer et al. [2006] to profile the ballistics of a DRC, and a clip of speech. Compressed signals were generated using the Cockos VST ReaComp plugin with an RMS level detector set to 5ms.

The test signal defined in Bitzer et al. [2006] is a 1kHz sinewave whose amplitude varies above and below a DRC's threshold. For this experiment, the DRC's threshold is set to -10dB, thus the sinewave's amplitude modulates from 0.25 to 1.0, and then later from 1.0 to 0.25. The attack time is set to 40ms, and the release time set to 200ms; therefore to ensure the full transients of the DRC can be observed, the sinewave has amplitude 0.25 for 500ms, then has amplitude 1.0 for 1000ms, and finally goes back to amplitude 0.25 for 500ms. The compression ratio is set to 10, knee-width set to 0dB, and the makeup gain set to -3dB. The construction of this signal allows for a simultaenous profiling of the algorithm's ability to match the VST DRC's characteristic attenuation as well as ballistics on a straightforward signal.

For a more complicated signal, a two second clip of a man speaking was compressed using a preset suggested by Curtis Judd [1]. This "light-touch" compression uses a fast attack of 3ms and slow release of 300ms, knee-width of 1.0 dB, and compression ratio 5.0. The threshold of

[1] "Compression for Dialogue Audio - Presets for Video Editors - Quick and Dirty," Accessed 18 April, 2023

Table 7.1: DRC parameters used in ReaComp VST and learned from gradient descent on ballistics profiling signal proposed in Bitzer et al. [2006].

	ReaComp	Learned Value
Threshold (dB)	-10.0	-11.7
Ratio	10.0	4.8
Makeup Gain (dB)	-3.0	-3.0
Knee Width (dB)	0.0	1.6
Attack Time (ms)	40.0	54.0
Release Time (ms)	200.0	268.0
Smoothing Time (ms)	-	31.1

Table 7.2: DRC parameters used in ReaComp VST and learned from gradient descent on speech signal.

	ReaComp	Learned Value
Threshold (dB)	-15.0	-20.5
Ratio	5.0	1.8
Makeup Gain (dB)	3.9	3.4
Knee Width (dB)	1.0	0.0
Attack Time (ms)	3.0	35.1
Release Time (ms)	300.0	37.6
Smoothing Time (ms)	-	13.2

-15.0dB and makeup gain of 3.9dB were selected to ensure DRC was applied to the signal, and that the output signal was normalized to the same amplitude of the input signal.

7.3 Initialization and Optimization

Because the methodology proposed in Section 7.1 can be implemented in an autodifferentiating framework such as Tensorflow, a gradient descent can be performed to optimize DRC parameters for a given dry/wet audio pair. The cost is calculated by passing a dry audio sample through the estimated DRC and measuring the distance between the estimated and target wet audio signal. Gradient descent is performed using the Adam method with an initial learning rate of 10^{-4} [Kingma and Ba, 2015]. The cost function chosen is MSS loss with window sizes 46ms, 12ms, and 3ms [Engel et al., 2019]. This loss function is chosen to avoid phase issues that may arise. Optimization is allowed to run for a maximum of 40000 iterations. Early stopping is employed with a patience of 1000 iterations.

The DDSP DRC parameters must be initialized such that each contributes to the compression applied to the dry signal. Otherwise, these parameters will not update during optimization. Furthermore, "reasonable" parameters should be chosen to avoid portions of the loss surface very far from expected DRC parameters. As such the threshold value is initialized close to the mean value of the dry signal's downsampled RMS level curve, the ratio initialized close to 2.0, the knee-width initialized close to 2dB, makeup gain initialized just above 0dB, τ_{at} and

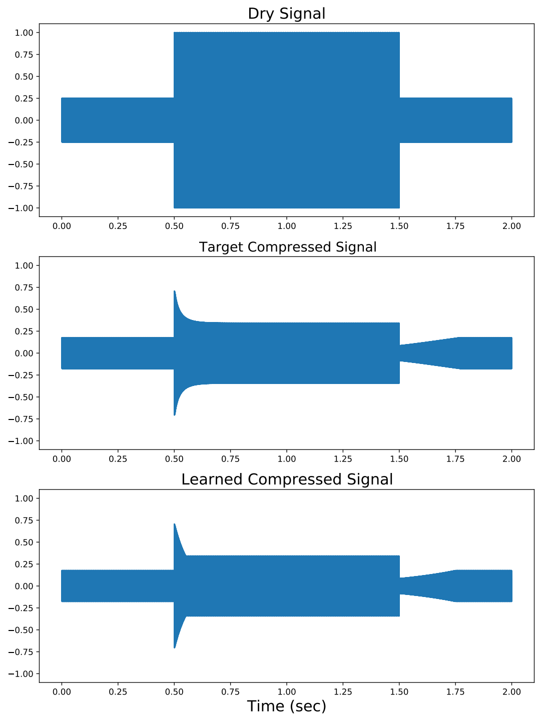


Figure 7.2: Waveforms for the dry, compressed, and reverse engineered test signals proposed by Bitzer et al. [2006]

τ_{st} initialized close to 45 frames (about 10ms), and τ_{rt} initialized close to 450 frames (about 100ms).

7.4 Results

Tables 7.1 and 7.2 compare the VST plugin settings to the parameters learned in the gradient descent. Figures 7.2 and 7.4 show the uncompressed, VST compressed, and differentiable DRC compressed waveforms for the test signal and speech signal respectively. Figures 7.3 and 7.5 show the downsampled loudness curves of the VST compressed and differentiable DRC compressed waveforms for the test signal and speech signal respectively.

7.5 Discussion

In general it is difficult to compare parameter settings across DRCs as their implementations vary greatly across designs. Furthermore, few DRCs have an explicit τ_{st} to measure. As such attention will be paid to the compressed waveforms themselves.

With the test signal, the differentiable DRC is able to match the static characteristic of the target signal well, with nearly 0dB residual. Though the learned release time is off by 68ms from the VST, the differentiable DRC's release passage closely matches that of the VST's. The biggest discrepancy occurs during the attack passage, where the differentiable DRC cannot match the VST's sloped attack attenuation settling.

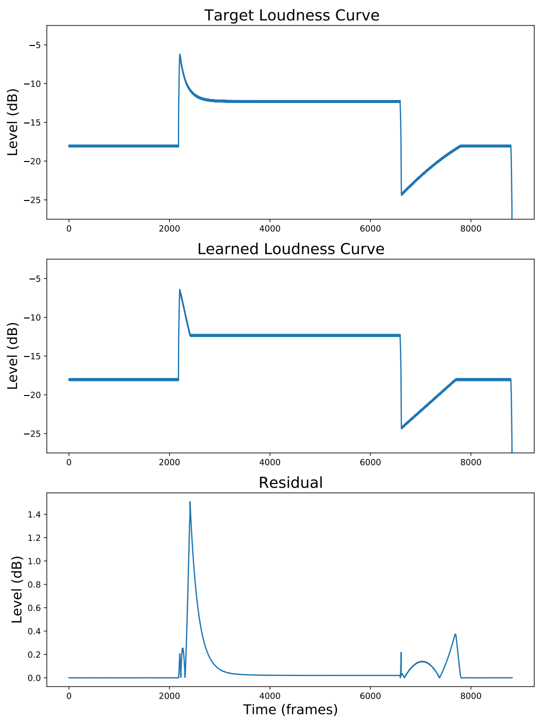


Figure 7.3: Loudness curves for the compressed test signal, the reverse engineered signal, and the residual between the two.

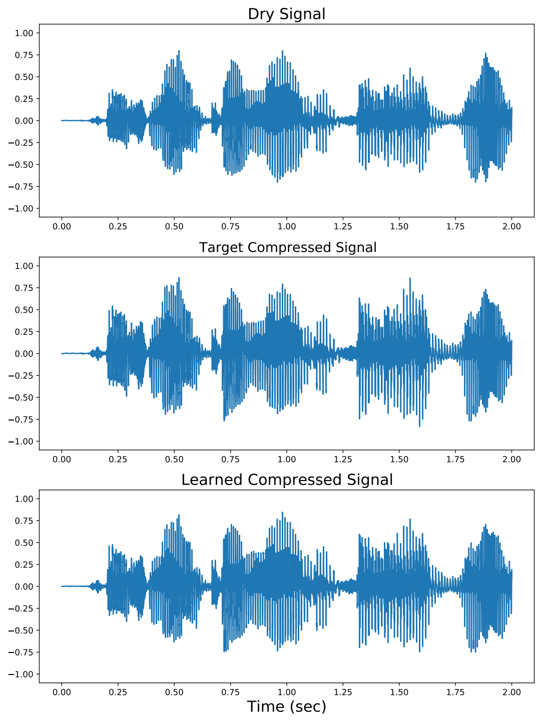


Figure 7.4: Waveforms for the dry, compressed, and reverse engineered speech signal.

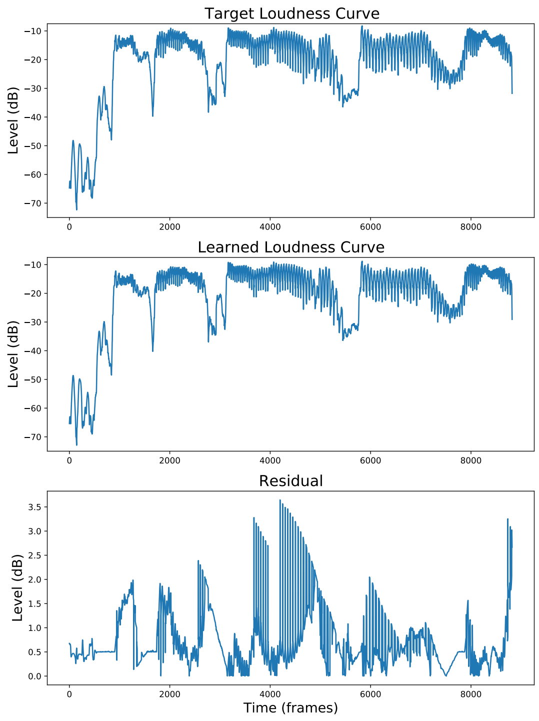


Figure 7.5: Loudness curves for the compressed speech signal, the reverse engineered signal, and the residual between the two.

The speech signal tasks the differentiable DRC with matching a "light touch" compression on a dynamic signal, which it is able to do within about 3.5dB. Though the differentiable DRC's threshold is set several dB lower than the VST, it compensates with a longer attack time — this smoothing decreases the initial attenuation to portions of the loudness curve just above -20dB. It is interesting to note that the differentiable DRC learns a makeup gain 0.5dB smaller than the target. Better initialization may help avoid this bias and improve the parameter estimation.

Though a formal listening test would need to be conducted to make any certain claims about the perceptual matching of these reverse engineered waveforms, some informed guesses can be made. According to a perceptual evaluation presented in Bromham et al. [2022], both novice and experienced listeners could only discern differences in compressed signals when attack differed from 10ms to 30ms, and when release differed from 100ms to 300ms. Furthermore, these differences in ballistics are more pronounced for higher threshold DRC settings than lower threshold settings. Therefore, it is doubtful that the reverse engineered DRC on the speech signal is perceptually indistinguishable from the reference. Given the closeness of the reverse engineered DRC's parameters to those used for the test signal, there may be more of a chance of perceptual matching.

Regardless, the algorithm presented here represents a step forward in DDSP modelling of DRC effects. As opposed to Steinmetz et al. [2022], the DRC here is able to separately model attack and release times, and as opposed to Wright et al. [2022], the DRC here implements no recursive calculations and thus avoids differentiation through time.

Chapter 8

Reverse Engineering a Linear Time-Invariant Multitrack Mix

The DDSP modules presented in Engel et al. [2019] are sufficient to recreate the multitrack linear effect matching of Barchiesi and Reiss [2010]. To briefly summarize Barchiesi and Reiss [2010], the authors choose LTI filters of order N and fit them to the effect processing applied to the individual tracks of a multitrack in the time domain. The fitted impulse response for each filter is then factored into a gain, delay, EQ, and panning parameter. A similar process could be implemented with the FIR EQ blocks presented in Engel et al. [2019], where a user could specify the order of the EQ module and fit the EQ parameters using gradient descent.

However, it is possible to improve on the approach presented in Barchiesi and Reiss [2010] in two ways just using the modules in Engel et al. [2019]. First, Engel et al. [2019] includes a learnable convolutional reverb effect which would expand the possible effects modelling of Barchiesi and Reiss [2010]. There still remains the question, though, of how best to incorporate reverb into the learned mixing chain, which is explored in this chapter. Second, the EQs learned in Barchiesi and Reiss [2010] are unconstrained, meaning there is no guarantee that the learned filters will be easy to engage with. As demonstrated in this chapter it is possible to constrain the EQ modules of Engel et al. [2019] to mimic an octave band graphic EQ, which increases interpretability and the possibility of intervention.

Thus in this chapter a method to retrieve the parameters used to create a multitrack mix using only raw tracks and the stereo mixdown is presented. This method is able to model linear time-invariant effects such as gain, panning, EQ, delay, and reverb. The optimization procedure used is gradient descent in the spectral domain, with the aid of differentiable digital signal processing modules. This method allows for a fully interpretable representation of the mixing signal chain by explicitly modelling audio effects using greybox DDSP modules, rather than using differentiable blackbox modules. The chapter is organized as follows:

- Sec 8.1 presents a formal problem statement to be solved.
- Sec 8.2 presents a formulation of each effect to be fitted in the mixing chain, methods for initializing their parameters, and the optimization procedure used to reverse engineer a mix. Two reverb module architectures are proposed, a "stereo reverb" model and an

"individual reverb" model.

- Sec 8.3 gives the results of an experiment consisting of reverse engineering six mixdowns mixed using only linear time-invariant effects processing. Objective measures are presented using MIR features, and a formal listening test is used to measure the perceptual closeness of the reverse engineered mixes to their reference mixdown.
- Sec 8.4 concludes by discussing the results

8.1 Formal Problem Statement

Let $y(n)$ represent a target mixdown, and let $y(\hat{n})$ represent the mixdown produced by some mixing chain characterized by a set of parameters θ. The goal is to find values θ that correspond to parameter settings in a mixing chain that will minimize $||y(n) - \hat{y}(n)||$, where $||\cdot||$ denotes some cost function.

8.2 Architecture and Optimization

8.2.1 Overview

The signal processing chain applied to each input raw track is as follows: Dry Input $\rightarrow$ FIR EQ $\rightarrow$ Gain $\rightarrow$ Pan $\rightarrow$ Reverb & Wet/Dry Mix $\rightarrow$ Sum with other stems. Note that since these effects are all linear time-invariant, the order of application of the effects is arbitrary. In mix engineering, a "stem" refers to a raw track that has had processing applied to it. To drive each module, a set of parameters Θ_{module} are estimated. For example, Θ_{EQ} refers to the set of parameters estimated to drive the EQ module. A stem which has been processed by applying both EQ and gain to a raw track $x(n)$ can be written as

$$\text{stem}_{EQ,\ Gain}(n) = \text{Gain}(\text{EQ}(x(n)|\Theta_{EQ})|\Theta_{Gain}) \tag{8.1}$$

and a stereo mixdown of N raw tracks $x_i(n)$ with EQ, gain, pan, and reverb applied can be written as

$$\begin{aligned} \hat{y}_L(n) &= \sum_{i=1}^{N} \text{Reverb}(\text{Pan}_L(\text{Gain}(\text{EQ}(x(n)_i|\Theta_{EQ})|\Theta_{Gain})|\Theta_{Pan})|\Theta_{Reverb}) \\ \hat{y}_R(n) &= \sum_{i=1}^{N} \text{Reverb}(\text{Pan}_R(\text{Gain}(\text{EQ}(x(n)_i|\Theta_{EQ})|\Theta_{Gain})|\Theta_{Pan})|\Theta_{Reverb}) \end{aligned} \tag{8.2}$$

8.2.2 EQ

The frequency transfer curve module is used for equalisation by multiplying an input signal's short-term Fourier Transform (STFT) magnitude response with a user specified curve in the frequency domain [Engel et al., 2019]. In this work, a 1025 point frequency transfer curve Θ_{EQ} is used. This corresponds to a finite impulse response (FIR) EQ with 2048 taps in its impulse response. Given a raw track $x(n)$, the EQ module can be written as

$$\text{EQ}(x(n)|\Theta_{EQ}) = \text{ISTFT}\big(\text{STFT}(x(n)) \times \Theta_{EQ}\big) \tag{8.3}$$

where ISTFT refers to the inverse short-time Fourier transform and $\times$ refers to pointwise multiplication.

In this work the EQ is modelled after a 10 band FIR graphical EQ [Välimäki and Reiss, 2016], which can be characterized using a 10 dimensional $\Theta_{EQ\ gains}$. The ten values specify the gain of each octave band filters, which are centered at 30, 60, 125, 250, 500, 1000, 2000, 4000, 8000, and 16000 Hz respectively. Shelving filters are used for frequencies below 30Hz and above 16000 Hz that match the attenuation specified at the lowest and highest octave band respectively.

The following procedure is used to calculate the 1025 dimensional Θ_{EQ} that will approximate a 10 band FIR graphical EQ. First a 10 dimensional $\Theta_{EQ\ gains}$ is generated. Then, these values are transformed via

$$\Theta_{EQ\ gains} \leftarrow 1 - \sigma(\Theta_{EQ\ gains}) \tag{8.4}$$

where σ denotes the sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{8.5}$$

The values in the transformed $\Theta_{EQ\ gains}$ range from $(0, 1)$ due to the bounds of the sigmoid function.

Finally a piecewise linear frequency transfer curve Θ_{EQ} is constructed using linear interpolation between the octave band attenuations specified by $\Theta_{EQ\ gains}$. Thus the EQ module's frequency transfer curve is bounded from (0,1) at all points The estimated values are initialized with random uniform noise from [-1,1], which initializes the octave band gains from -6dB to -1dB.

8.2.3 Gain and Pan

The gain module is formulated as

$$\text{G}(x(n)|\Theta_{Gain}) = \text{LReLU}(\Theta_{Gain}) \times x(n) \tag{8.6}$$

where LReLU() refers to the leaky rectified linear unit [Maas et al., 2013]

$$\text{LReLU}(x) = \begin{cases} \beta x, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{8.7}$$

with tunable parameter β. For this work $\beta = 0.5$ has been chosen. Note that these gains can go negative, which corresponds with applying a phase shift to the EQed stem. The gain parameters Θ_{Gain} are initialized with random uniform noise from [0.9,1.1], which corresponds to gains from -0.915dB to 0.828dB.

The pan module utilises a linear panning law and is formulated as

$$\begin{aligned} \text{Pan}_L(x(n)|\Theta_{Pan}) &= \big(0.5 + (0.5 \times \tanh(\Theta_{Pan}))\big) \times x(n) \\ \text{Pan}_R(x(n)|\Theta_{Pan}) &= \big(1 - \text{Pan}_L\big) \times x(n) \end{aligned} \tag{8.8}$$

where $\times$ denotes pointwise multiplication and tanh() denotes the hyperbolic tangent function

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{8.9}$$

The panning module applies a gain of Pan_L to the signal before sending it to the left channel and a gain of Pan_R to the signal before sending it to the right channel. The pan parameters Θ_{Pan} are initialized with mean 0, variance 10^{-6} Gaussian noise.

8.2.4 Reverb

Similar to the EQ module, the reverb module also performs convolution with a given impulse response via multiplication in the frequency domain. Instead of estimating a frequency transfer curve, however, the reverb module directly estimates an impulse response.

Two reverb architectures were tested in this work, one using a stereo reverb bus (thus requiring two estimated impulse responses for a mixdown), and one using two impulse responses per channel (thus requiring [2 x Number of Tracks] impulse responses for a mixdown). Figures 8.1 and 8.2 show block diagrams for the two architectures respectively.

For the stereo reverb bus architecture, a wet/dry mix is produced by performing a weighted sum with the signal input to the reverb module with the signal output by the reverb module. Thus the module estimates θ_{IR} for the reverb's impulse response and $\theta_{W/D}$ for the module's wet/dry mix. For the individual bus architecture, $\theta_{W/D}$ is omitted.

In the stereo reverb bus case, the module's output is formulated as

$$output = dry + \sigma(\theta_{W/D}) \times (dry * \theta_{IR}) \tag{8.10}$$

where $\sigma()$ denotes the sigmoid function. Note that should bypass=0, the module applies no reverb and passes the dry signal through. With bypass=1, this module applies a convolutional reverb to an input and sums it in proportion to the dry input. $\theta_{W/D}$ is initialized with uniform random noise from $[-0.3, 0.3]$, which corresponds to a range of -7 dB to -5 dB.

In the individual reverb bus case, the output becomes

$$output = dry + dry * \theta_{IR} \tag{8.11}$$

where $*$ denotes the convolution operation. In both cases, θ_{IR} is initialized with mean 0, variance 10^{-6} random Gaussian noise.

The choice to investigate two reverb architectures stems from a desire to balance the method's modelling capacity with network size and complexity. Mathematically speaking, both a left and right convolutional reverb impulse response must be estimated for each raw track in a multitrack in order to fully characterize the mixing chain. This is necessary because mixing engineers typically use stereo reverb impulse responses that are decorrelated in the left and right channel to increase spatialisation [Kendall, 1995].

However, the number of learned reverb parameters θ_{IR} is orders of magnitude larger than $\theta_{W/D}$, θ_{EQ}, θ_{Pan}, and θ_{Gain} combined. As formulated in this paper, the number of parameters needed to describe the mixing chain before the reverb module is 12: one for gain, one for pan, and 10 for the graphic EQ. In order to model a one second convolutional reverb impulse response sampled at CD quality, $44,100$ values are needed. Given that multitracks often contain more than 10 raw tracks, at least 20 impulse responses would have to be estimated for a full characterization, which balloons θ_{IR} to $882,000$ estimated values. The stereo reverb model would cap θ_{IR} at $88,200$ parameters in this case, regardless of how many raw tracks make up the multitrack. In this formulation $\theta_{W/D}$ is necessary to control the "amount of reverb" applied to each raw track sent to the left and right channels.

8.2.5 Loss and Optimization Procedure

Given randomly initialized Θ_{gain}, Θ_{pan}, Θ_{EQ}, Θ_{IR}, and $\Theta_{W/D}$, target mixdown $y(n)$, raw tracks $x_i(n)$ and estimated mixdown $\hat{y}(n)$ as described in Eqn 8.2, stochastic gradient descent can be used to minimize $||y(n) - \hat{y}(n)||$ by updating the module parameters Θ, where $|| \cdot ||$ denotes some norm used as a cost function.

In this work, a multi-scale spectrogram (MSS) loss is used as the cost function $|| \cdot ||$ [Engel et al., 2019], which was inspired by the multi-resolution spectral amplitude distance demonstrated in Wang et al. [2020]. As the name implies, MSS computes a norm by measuring the distance between the spectrograms of two audio signals with varying STFT window sizes and performing a weighted sum of these differences. Though mean absolute error (MAE) in the time domain is often used in audio applications and is cheaper to compute than MSS loss, the latter was chosen because it ignores the phase differences between the target and estimated signals, which mimics human perception [Chi et al., 2005]. The resolutions for the spectrograms used are 2048, 512, and 128 samples. At 44.1kHz sampling rate, these correspond to windows of size 50ms, 12ms, and 3ms. An L1 loss is computed on these spectrograms, which is the absolute value of the difference between the spectrograms reduced across both the frame and frequency dimensions.

Stochastic gradient descent was performed with learning rate scheduling and early stopping [Darken et al., 1992]. The descent begins using the ADAM optimizer with learning rate 10^{-3} [Kingma and Ba, 2015]. Once the loss reaches an early stopping criterion, the learning rate is dropped to 10^{-4}. After the same procedure happens again, the learning rate is further dropped to 10^{-5}. The gradient descent concludes thereafter.

8.3 Results and Analysis

All audio used in this work is sampled at 44.1kHz, corresponding to CD quality audio. A professional mixing engineer was tasked with producing mixdowns for five separate multitrack recordings. The multitracks were chosen because they ranged from roughly 10 to 20 raw tracks each, had representative excerpts between 20 to 30 seconds in length, and were diverse in instrumentation and genre. All stereo tracks were converted to two mono tracks. All multitracks were downloaded from the Cambridge Multitracks dataset [Senior, 2011]. Table 8.1 shows the artist and song title for each song used in this work. For each multitrack, three approximated mixdowns were calculated: a stereo bus approximation (2 Bus), an individual bus approximation (Ind. Bus), and a stereo gain mix approximation using the least squares estimatation method in Barchiesi and Reiss [2010] (Gain Mix).

Similar to the procedure followed in Barchiesi and Reiss [2010], the mixing engineer only used linear processing to create the mixdown, including gain, pan, EQ, delay, and reverb. No distortion or dynamic range compression was used. Moreover no automation was used on the linear effects. All audio discussed in this paper can be found at `https://jtcolonel.github.io/RevEng/`.

Table 8.1: Songs chosen for mixing and shortened names used in this paper.

Artist	**Song Name**	**Genre**	**Reference Name**
Araujo	The Saga of the Harrison Crabfeathers	Jazz	Saga
Blue Lit Moon	Dad's Glad	Alternative Rock	Glad
Carol Dant	I am the Desert	Electronica	Desert
The Complaniacs	Etc	Punk	Etc
Timboz	Pony	Metal	Pony

Table 8.2: Average relative errors by feature subgroup for approximated mixdowns compared to the reference mixdown.

Mixdown	**Spectral**	**PMF**	**Stereo**	**Loudness**	**Total**
Desert Two Bus	0.95%	14.86%	5.99%	1.68%	4.88%
Desert Gain Mix	5.87%	15.96%	13.00%	4.80%	9.21%
Desert Ind Bus	35.55%	104.67%	133.84%	25.24%	70.52%
Etc Two Bus	2.52%	6.57%	14.07%	3.67%	6.54%
Etc Gain Mix	19.21%	95.99%	33.83%	5.94%	32.91%
Etc Ind Bus	9.63%	62.78%	20.37%	22.64%	25.07%
Glad Two Bus	7.10%	6.04%	32.49%	1.80%	12.16%
Glad Gain Mix	23.16%	38.72%	51.85%	8.76%	29.60%
Glad Ind Bus	34.83%	91.09%	92.46%	17.52%	55.13%
Pony Two Bus	1.70%	21.34%	9.95%	7.67%	8.82%
Pony Gain Mix	28.24%	49.42%	35.33%	15.68%	30.49%
Pony Ind Bus	61.16%	55.86%	159.22%	16.65%	74.21%
Saga Two Bus	2.25%	3.26%	3.53%	0.41%	2.28%
Saga Gain Mix	13.57%	25.52%	17.44%	12.03%	16.26%
Saga Ind Bus	27.37%	356.85%	137.33%	61.17%	122.17%

8.3.1 Objective Evaluation

To objectively measure how close the estimated mixdowns matched the target mixdowns, a set of low-level audio features were calculated and compared according to the methodology in Wilson and Fazenda [2016]. These features can be subgrouped into spectral measures [Bogdanov et al., 2013], loudness measures [Recommendation], stereo features [Tzanetakis et al., 2007], and envelope probability mass function (PMF) features [Wilson and Fazenda, 2014]. The PMF features are calculated by making a histogram of the values a digital audio

Table 8.3: F stat and p-values for each multitrack and all multitracks, including all participants.

Group	F Stat	p-value
All Songs	316.178	3.737×10^{-107}
Desert	143.595	4.127×10^{-32}
Etc	100.860	4.195×10^{-27}
Glad	91.140	9.891×10^{-26}
Pony	70.035	2.575×10^{-22}
Saga	42.554	1.529×10^{-16}

Table 8.4: F stat and p-values for each multitrack and all multitracks, including only participants with audio production or mixing experience.

Group	F Stat	p-value
All Songs	207.896	1.322×10^{-57}
Desert	47.444	6.977×10^{-12}
Etc	91.529	8.701×10^{-16}
Glad	117.997	2.208×10^{-17}
Pony	53.478	1.450×10^{-12}
Saga	35.129	3.062×10^{-10}

signal takes, normalizing this historgram so it becomes a probability mass function, and then calculating statistical measures of this PMF.

To aggregate each mixdown's performance, the average relative error of each subgroup of features can be calculated. Relative errors must be taken as different features have different units of measurements, and furthermore different feature measures can be orders of magnitude apart from one another. Observing an average relative error across subgroups of features allows for an overall picture of how each mixdown matched the reference across perceptual correlates.

Table 8.2 provides a summary of the average errors across each feature subgroup for each approximated mixdown relative to the reference mixdown. Within each song, the two bus architecture outperforms both the individual bus architecture and gain mix across all subgroups of features. For the songs "Saga," "Pony," "Desert," and "Dad's Glad," the gain mix outperforms the individual bus mix in average relative error across each subgroup of features. For the song "Etc," the individual bus mix outperforms the gain mix in average relative error for spectral, PMF, and stereo measures, with the gain mix performing best for the loudness features.

There are certain instances where the gain mix or the individual bus mix matches the reference more closely than the two bus mix in a given feature. When measuring 95% spectral rolloff, the individual bus mix performs best in the song "Saga." When measuring spectral spread, the gain mix outperforms the two bus mix in the songs "Dad's Glad" and "Saga." These are the only instances in spectral measure where the two bus architecture does not perform best.

When measuring PMF centroid, the gain mix performs best for the song "Dad's Glad," and the individual bus performs best for the song "Etc." For PMF skew, the gain mix performs best in the songs "Desert" and "Pony." For PMF kurtosis, the gain mix performs best in the song "Desert." In all other PMF measures, the two bus architecture performs best.

Table 8.5: Results of pairwise comparison of mixdown architecture on perceptual similarity rating across multitracks, with Bonferroni Correction, o $>$ 0.05, *<0.001 ·=no comparison

	Reference Mix	Gain Mix	Ind Bus Mix	2 Bus Mix
Reference Mix	·	*	*	o
Gain Mix	*	.	*	*
Ind Bus Mix	*	*	.	*
2 Bus Mix	o	*	*	.

Table 8.6: Results of pairwise comparison of mixdown architecture on perceptual similarity rating within the "Etc" multitrack, with Bonferroni Correction, o $>$ 0.05, *<0.001 ·=no comparison

	Reference Mix	Gain Mix	Ind Bus Mix	2 Bus Mix
Reference Mix	·	*	*	*
Gain Mix	*	.	*	*
Ind Bus Mix	*	*	.	*
2 Bus Mix	*	*	*	.

When measuring both the stereo panning spectrum across all bands and across high frequency bands, the individual bus mix performs best for the songs "Dad's Glad" and "Etc." For the mid band stereo panning spectrum, the individual bus performs best for the song "Etc." For the stereo left-right ratio, the gain mix performs best for the song "Desert." In all other stereo measures, the two bus architecture performs best.

For the loudness range measure, the gain mix performs best for the song "Pony." For the average crest factor measured with a resolution of 100ms, the individual bus mix performs best for the song "Desert," and the gain mix performs best for the song "Pony." When the resolution of the crest factor is increased to 1000ms, the gain mix performs best for the song "Desert." The two bus mix performs best for all other measures of loudness.

8.3.2 Perceptual Evaluation

While much research has been done to numerically characterize timbre (see Peeters et al. [2011]) and relate closeness of timbres within a perceptual space (e.g. Caclin et al. [2005],Elliott et al. [2013]), there are no perfect numerical measures for determining how close two timbres are. Furthermore, the timbral complexity of multitrack mixes renders numerically characterizing mixes challenging [Wilson and Fazenda, 2015, Colonel and Reiss, 2019]. While the results presented in the objective evaluation show the two bus mix outperforming the other two mixes in numerical measures, this is no guarantee that the two bus mix sounds most similar to the reference mix. Thus, a listening study was performed to assess how well each method perceptually matched the reference mix.

Participants were presented with the following text when beginning the listening test:

> Multitrack mixing describes the process of combining distinct pieces of audio into a final 'mixdown.' This process often includes editing and applying sound effects to a set of multitrack recordings to produce a mixed song.
>
> This listening test will present you with a reference mix which has been mixed by

Table 8.7: Results of pairwise comparison of mixdown architecture on perceptual similarity rating within the “Pony” multitrack, with Bonferroni Correction, o > 0.05, *<0.001 ·=no comparison

	Reference Mix	Gain Mix	Ind Bus Mix	2 Bus Mix
Reference Mix	·	*	*	o
Gain Mix	*	·	o	*
Ind Bus Mix	*	o	·	*
2 Bus Mix	o	*	*	·

Table 8.8: Results of pairwise comparison of mixdown architecture on perceptual similarity rating within the “Desert” multitrack, including only participants with audio production experience with Bonferroni Correction, o > 0.05, *<0.001 ·=no comparison

	Reference Mix	Gain Mix	Ind Bus Mix	2 Bus Mix
Reference Mix	·	o	*	o
Gain Mix	o	·	o	*
Ind Bus Mix	*	o	·	*
2 Bus Mix	o	*	*	·

> a professional audio engineer. You will be asked to rate different mixes according to the similarity of these in relation to the reference mix. Please rate the sounds you hear from 'least similar' to 'most similar,' using the full scale when possible. Each mix will be of medium duration: 25-30 seconds.
>
> The aim of the test is to identify which mix is closer to the reference example. This will be done for different songs, each of which has been mixed using a unique combination of processing and effects. The test will take approximately 25 minutes and should be conducted on headphones.

23 participants took part in a listening study to evaluate the approximated mixdowns of the five multitracks. 14 participants reported having no audio production or mixing experience, while nine participants reported having some audio production or mixing experience. Demographic information was not collected of the participants, which was an oversight by the author. The choice to recruit both novice and experienced listeners was to ensure that a large enough sample size could be taken to perform statistical analysis on the listening test results. However, given the larger than expected response by experienced listeners, their results will be given a separate treatment from the whole population.

Participants were tasked with rating a set of mixes based on how closely the mixes matched a reference mix on a continuous scale of 0 to 1, where 0 represented a mix “very far” from the reference, and 1 represented a mix “matching exactly” the reference mix.

During the test participants were presented each of the mulitracks in a random order, one-by-one. For a given multitrack, participants were presented with four stimuli to rate against the reference mix. One stimulus was the identical reference mix. The other presented stimuli were three approximated mixdowns: a stereo bus approximation, an individual bus approximation, and a stereo gain mix approximation. Participants evaluated mixes for each of the five multitracks, thus providing a total of 20 ratings. Participants were encouraged to use the full 0 to 1 rating range when appropriate. Furthermore participants were given no time limit for the test, and no limit was placed on how many times a participant could listen to a given stimulus.

The analysis that follows is adapted from the perceptual study presented in Moffat and Reiss [2018]. The null hypothesis is that the perceptual evaluation scores are from the same distribution. A one-way ANOVA, with Bonferroni correction, shows for all mixdowns that the effect the method used to reverse engineer the mix had on user perception was statistically significant. This result holds when analysing the ratings separated by each multitrack as well. Table 8.3 lists the f stats and p-values. Figures 8.3 and 8.4 show the box plots of the overall results of the listening study.

With the null hypothesis rejected, a post-hoc Tukey pairwise comparison, with Bonferroni correction to reduce the chance of type I errors, was used. Table 8.5 shows the results of these pairwise comparisons for all architectures used. The pairwise comparisons demonstrate that the mean of participants' ratings for the reference mix and the stereo bus mix do not differ significantly. All other pairwise comparisons do differ significantly.

When breaking down the data by song, the above results hold for three of the five multitracks: "Blue," "Desert," and "Saga." For the multitrack "Etc," the pairwise comparisons demonstrate that the mean of participants' ratings differ significantly for all pairs including the reference and stereo bus model as shown in Table 8.6. For the multitrack "Pony," the reference/stereo bus model pair and gain mix/individual bus model do not have participant rating distributions that differ significantly as shown in Table 8.7.

Most of these results hold when excluding the listeners with no audio production or mixing experience. Box plots can be found in Figs 8.5 and 8.6. A one-way ANOVA, with Bonferroni correction, shows for all mixdowns that the effect the method used to reverse engineer the mix had on user perception was statistically significant. This result holds when analysing the ratings separated by each multitrack as well. Table 8.4 lists the f stats and p-values for this subset of participants with some audio production or mixing experience.

Post-hoc Tukey pairwise comparisons of the ratings provided by this subset of participants are similar to that of the whole group. When comparing across all songs and mixdowns, the values of the Tukey analysis match those presented in Table 8.5. This also remains true for the songs "Blue" and "Saga." For the multitrack "Etc," the pairwise comparisons again demonstrate that the mean of participants' ratings differ significantly for all pairs including the reference and stereo bus model as shown in Table 8.6. And again in "Pony," the reference/stereo bus model pair and gain mix/individual bus model do not have participant rating distributions that differ significantly as shown in Table 8.7. This subset's ratings do differ for the song "Desert," where there is no statistical difference between the stereo bus/reference pair and gain/reference pair. Results are shown in Table 8.8 As Fig 8.3 demonstrates, participants rated the stereo bus model as nearly identical to the reference mix, then rated the gain mix next closest, and finally rated the individual bus mix the furthest from the reference.

8.4 Discussion

The results of both the objective evaluation and the listening test suggests that the stereo reverb bus model outperforms both the gain mix and individual bus model in a reverse engineering of a mix task. Furthermore, the gain mix outperforms the individual bus model in almost all cases.

It is interesting to note that the individual bus architecture performs worse than a gain mix in both objective and subjective measures, given that the gain mix does not apply EQ or reverb. Even with the explicit ability to modify a raw track's spectral content, the individual bus architecture does a worse job than the gain mix of matching the reference mix's spectral features in four of five songs. "Etc" is the only song where the individual bus model outperforms the gain mix in spectral measures, and incidentally is the only song where the individual bus mix outperforms the gain mix in all other measures as well. Yet the results of the listening test place the individual bus estimate of "Etc" lower than the gain mix, for both experienced and inexperienced listeners. This highlights the difficulty in using objective features to characterize multitrack mixes. In its estimate the individual bus architecture placed a prominent reverb on the vocal stem that does not match the reference mix, which is most likely why it performed so poorly in the listener evaluation. In general, the mixes estimated by the individual bus architecture frequently apply much more reverb than the reference.

Across all tests the "Desert" mix from the individual bus model performed the worst in the listening test, and the "Desert" mix from the stereo bus model performed the best. In "Desert," several synthesizers (Synth1-6) are layered within the composition and provide backing to a layered vocal (Vox1-3). The reference mix applies both delay and reverb to most of the song's elements, in keeping with a "washed-out electronica" style mix, as well as EQ, gain, and pan. Figs 8.7 and 8.8 show Θ_{IR} and the frequency transfer curves for select stems from the song "Desert" produced by the individual bus model and stereo bus model respectively. Observing Figure 8.7, one can see the stark differences between the learned reverb impulse responses across the stems. Synth3 appears to have both echo and reverb applied, Synth6 has a dense reverb with significant energy in the tail applied, and Vox2 has a light echo applied. Note as well that the gradient descent produces distinct reverbs for the left and right channels for each of these stems. When listening to this mix, however, the vocals are barely audible, and the synthesizers dominate the mix. Figure 8.8 shows that the stereo bus model has combined both a reverb with less energy in the tail than the individual bus model with a prominent echo into the left and right channel learned impulse responses. The result is a mixdown that is nearly indistinguishable from the reference mix.

This failure by the individual bus model may be due to the relatively large parameter space the optimization has to navigate. It may be the case that given random parameter initialization for the 20 raw tracks, the optimization begins too far away from the parameters of the reference and instead converges to a random local minimum. Note that with 20 raw tracks, Θ_{IR} consists of 1.764×10^6 parameters. It is interesting to note that the individual bus model's optimization also does not match the reference when panning certain raw tracks. While the stereo reverb bus model pans Vox2 nearly center with $Pan_L = 0.502$, the individual reverb bus model pans Vox2 to the right with $Pan_L = 0.314$. This again suggests that the individual reverb bus model is exploring some area of the parameter space distant from the reference mix.

Future improvements may be made to the individual bus model by bypassing the reverb module for the beginning of the stochastic gradient descent. For example, the bypass could be activated until the first early stopping, which would allow for the network to best fit gain, pan, and EQ parameters before attempting to apply reverb. A full study of how DDSP performs in reverb estimation may also shed light on the issues of the individual bus model, including how a gradient descent performed on an FIR reverb IR can match IIR reverb implementations and reverb impulse responses with non-integer delays.

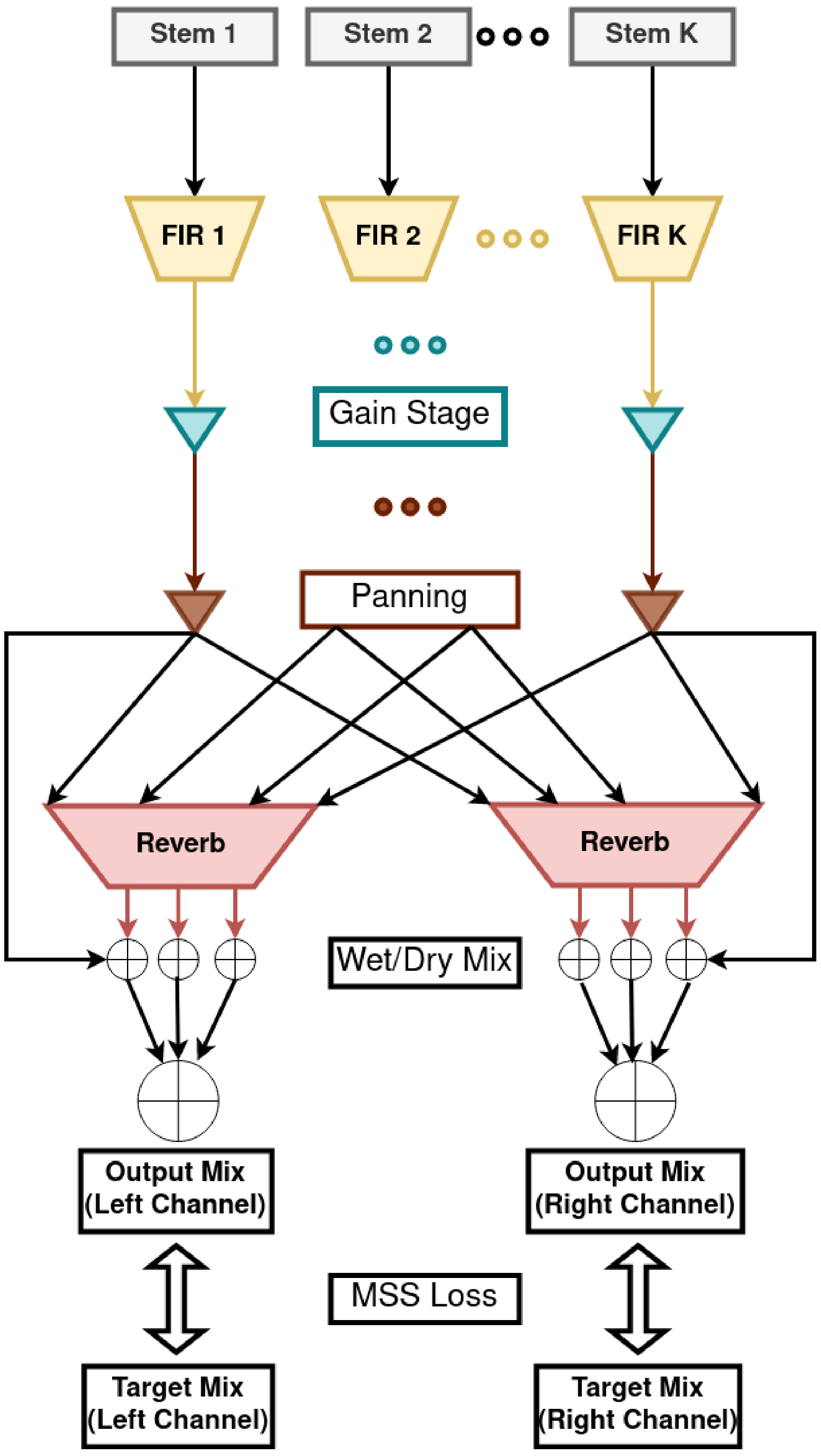

Stem 1
Stem 2
Stem K
FIR 1
FIR 2
FIR K
Gain Stage
Panning
Reverb
Reverb
Wet/Dry Mix
Output Mix (Left Channel)
Output Mix (Right Channel)
MSS Loss
Target Mix (Left Channel)
Target Mix (Right Channel)

Figure 8.1: Mixing chain diagram for the "stereo bus" architecture. In this architecture, only two reverb IRs are reverse engineered, regardless of how many stems are present in the multitrack.

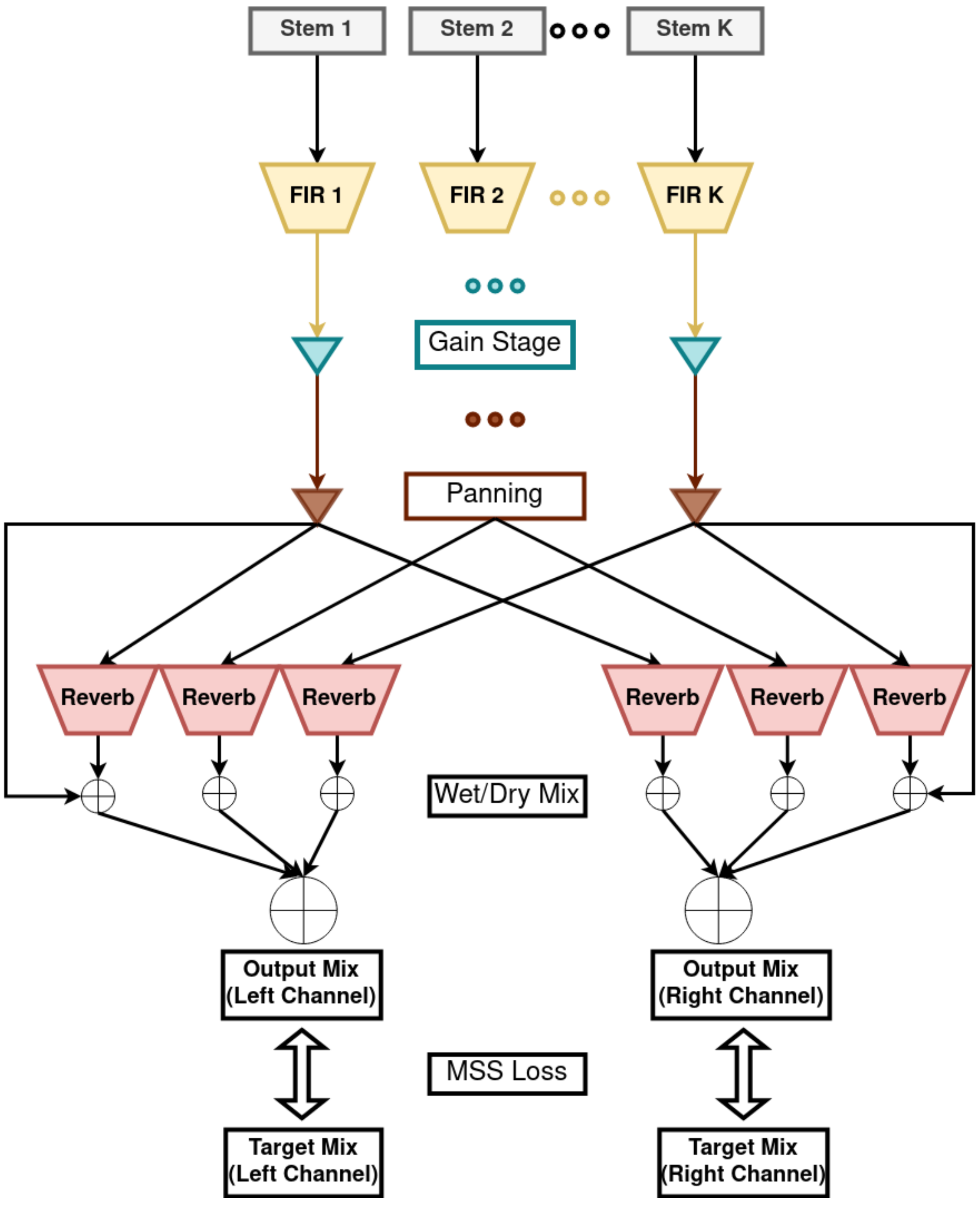


Figure 8.2: Mixing chain diagram for the "individual bus" architecture. In this architecture, each stem learns its own left and right channel reverb IRs. Thus in this diagram, a total of $2K$ reverb IRs are reverse engineered.

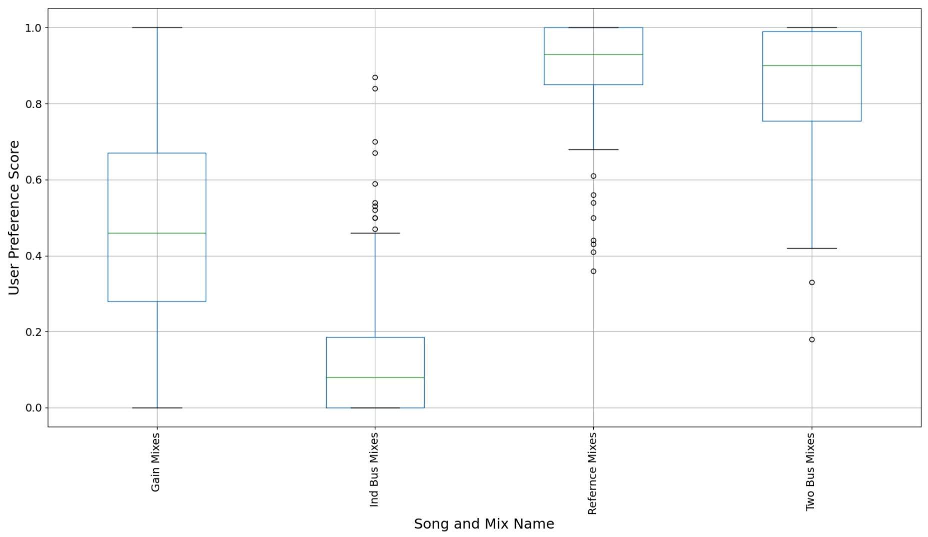


Figure 8.3: Box and whisker plots of the median, standard deviation and 95% confidence intervals of listener ratings broken down by reverse engineering architecture. The mixdown reverse engineered using the stereo reverb bus architecture was found to be perceptually indistinguishable from the reference.

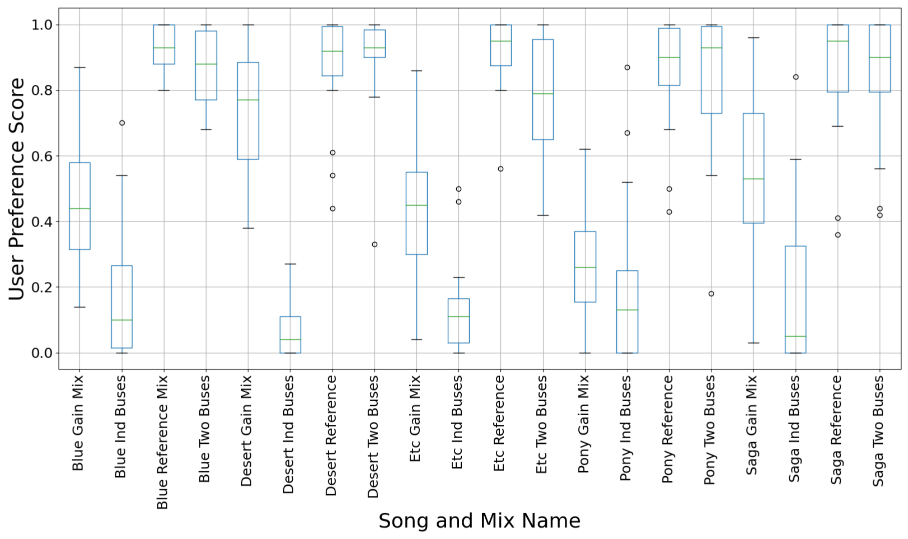


Figure 8.4: Box and whisker plots of the median, standard deviation and 95% confidence intervals of listener ratings broken down by mixdown. A full breakdown of results is in Sec 8.3.2

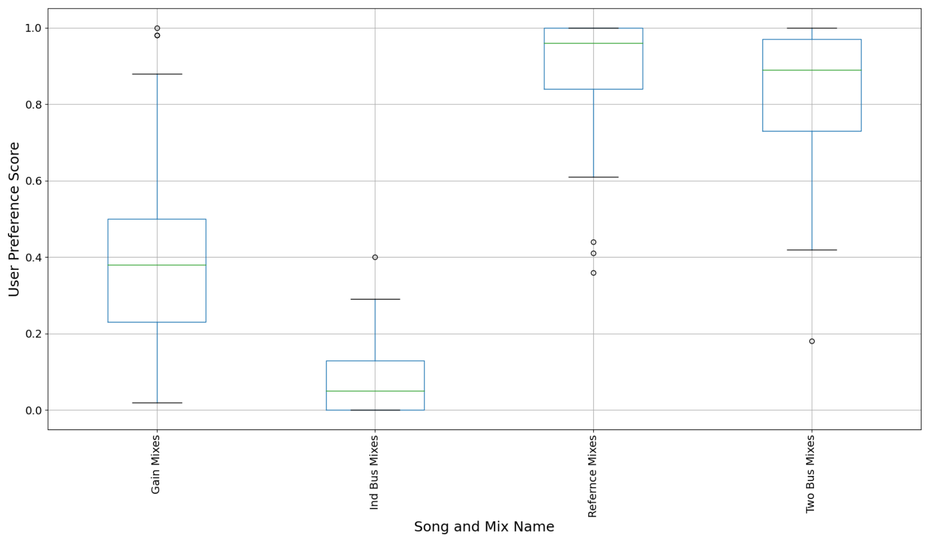


Figure 8.5: Box and whisker plots of the median, standard deviation and 95% confidence intervals of listeners with audio production experience ratings broken down by architecture. Within this subset of listeners it was again found that the mixdowns reverse engineered using the stereo bus reverb architecture are perceptually indistinguishable from a reference.

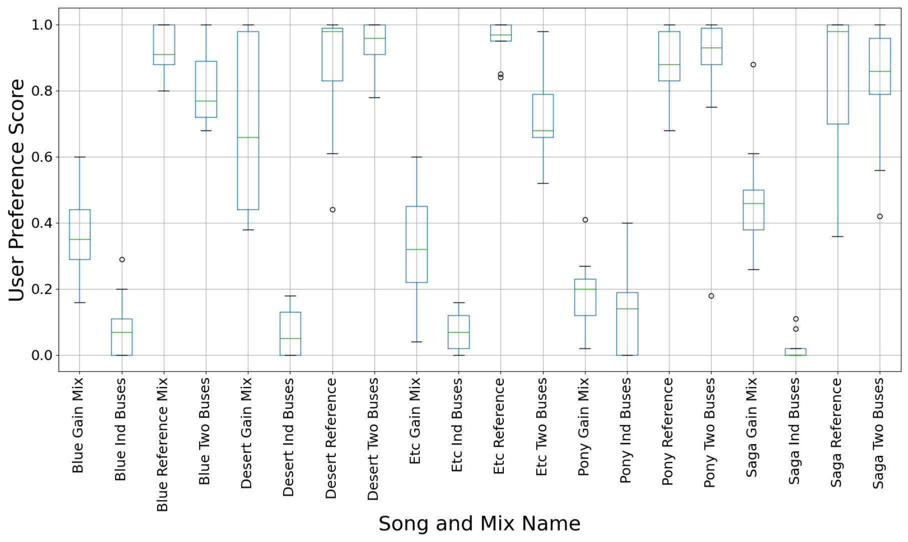
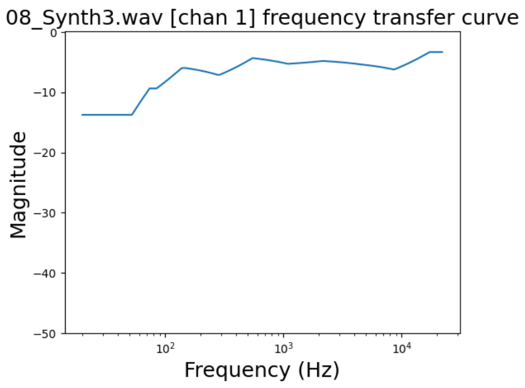
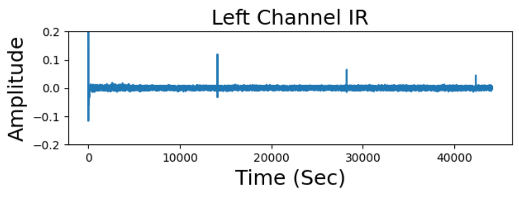


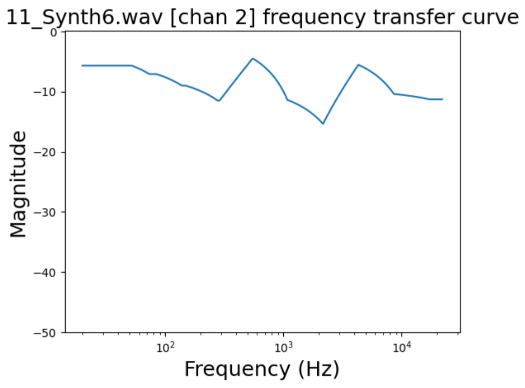
Figure 8.6: Box and whisker plots of the median, standard deviation and 95% confidence intervals of listeners with audio production experience ratings broken down by mixdown. A full breakdown of results is in Sec 8.3.2



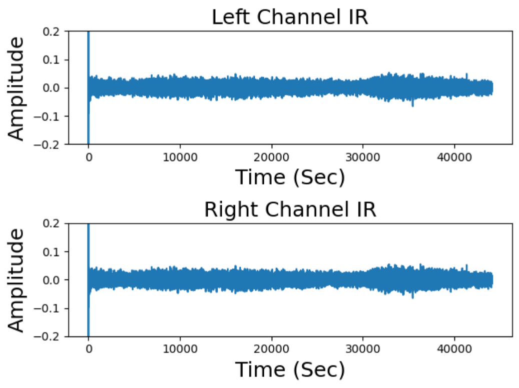
(a) Synth 3 EQ



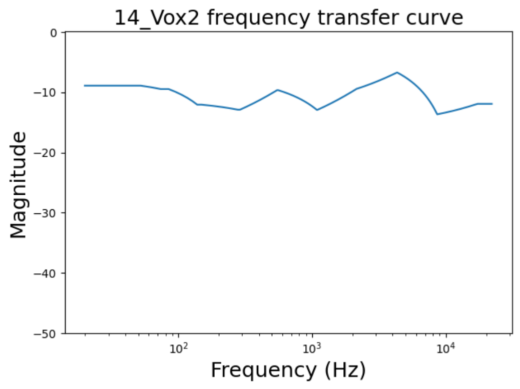
(b) Synth 3 Reverb IRs



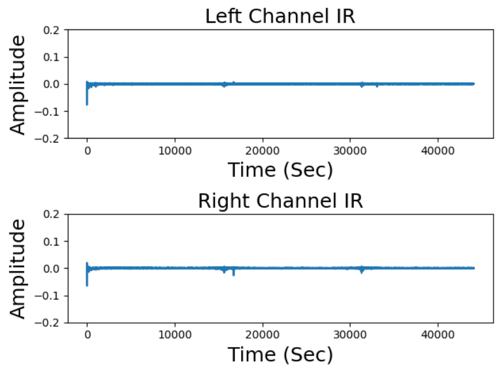
(c) Synth 6 EQ



(d) Synth 6 Reverb IRs

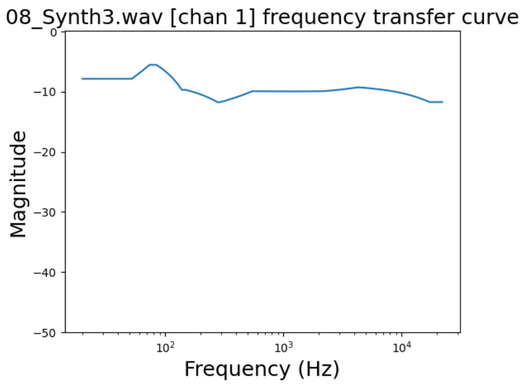


(e) Vocal 2 EQ

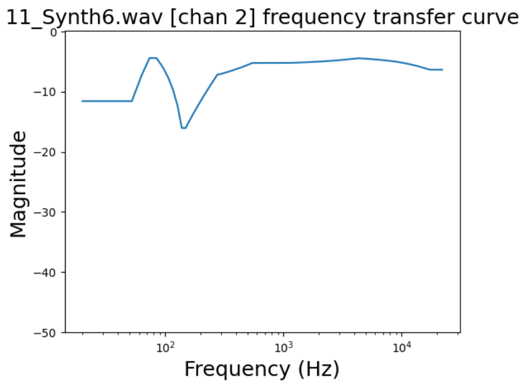


(f) Vocal 2 Reverb IRs

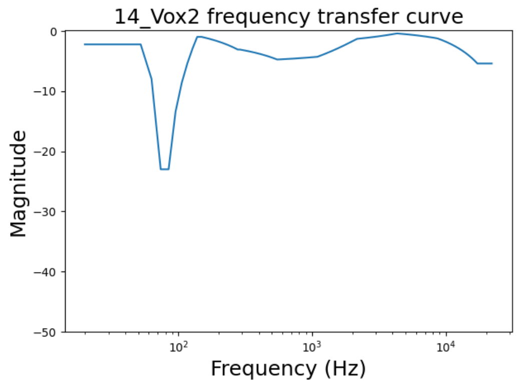
Figure 8.7: Learned EQs and reverbs for "Desert" using the individual bus model. The individual bus model learns an individual reverb IR for each stem, thus six reverb IRs are presented for three stems. In this case a very dense reverb IR with little decay was learned for Synth6, a separate dense reverb with delay for Synth3, and practically no reverb for Vox2.



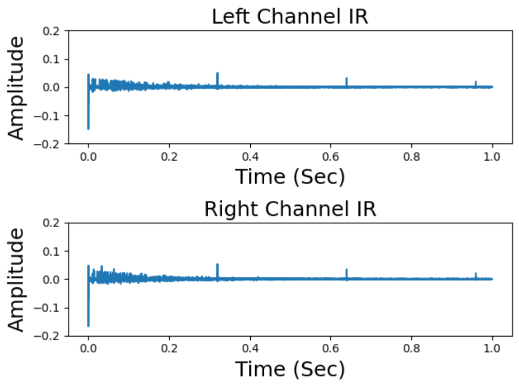
(a) Synth 3 EQ



(b) Synth 6 EQ



(c) Vocal 2 EQ



(d) Stereo bus reverb IRs

Figure 8.8: Learned EQs and reverbs for "Desert" using the stereo bus model. The stereo bus model learns a left and right reverb IR for the whole mixdown, thus two reverb IRs are presented for three stems. A reverb IR with reasonable decay and echo was learned for the mixdown.

Chapter 9

Reverse Engineering a Nonlinear Multitrack Mix

Though the results of Chapter 8 are heartening in their ability to reverse engineer a mix within perceptual tolerance, the constraint of only mixing with LTI processing is rather strict. Thus, to expand the reverse engineering methodology, the DDSP distortion presented in Chapter 6 and the DDSP DRC presented in Chapter 7 will be included in the "stereo reverb bus" mixing chain shown in Chapter 8.

This chapter seeks to synthesize the whole dissertation and answer the questions posed in Sec 1.2. A method for jointly optimizing the linear and nonlinear processing of a multitrack mix will be presented. By measuring the performance of this method by using a dataset of mixes made by students, there are fewer restrictions on the type of effects processing used to produce the dataset than in Barchiesi and Reiss [2010] or in Chapter 8. The reverse engineered mixing chains will explicitly model gain, panning, EQ, reverberation, memoryless distortion, and DRC using greybox DDSP modules. The parameters of these modules are formulated to be familiar to any mix engineer. Ultimately, a formal listening test is run in conjunction with an ablation study to determine which audio effects need to be reverse engineered in order to reach perceptual tolerance.

The chapter is organized as follows:

- Sec 9.1 repeats a formal problem statement to be solved for consistency's sake.
- Sec 9.2 presents a formulation of each effect to be fitted in the mixing chain, methods for initializing their parameters, and the optimization procedure used to reverse engineer a mix.
- Sec 9.3 describes how the multitracks were selected and how the mixdowns were produced that are used to evaluate the reverse engineering method.
- Sec 9.4 gives the results of an experiment consisting of reverse engineering the mixdowns. Objective measures are presented using MSS loss, and a formal listening test is used to measure the perceptual closeness of the reverse engineered mixes to their reference mixdown.

- Sec 9.5 concludes by discussing the results and demonstrating how the approach can be used to further research in perceptual correlates of mix engineering

9.1 Formal Problem Statement

Let $y(n)$ represent a target mixdown, and let $\hat{y(n)}$ represent the mixdown produced by some mixing chain characterized by a set of parameters θ. The goal is to find values θ that correspond to parameter settings in a mixing chain that will minimize $||y(n) - \hat{y}(n)||$, where $|| \cdot ||$ denotes some cost function.

9.2 Architecture and Optimization

All raw tracks and mixdowns are sampled at 44.1kHz. The signal processing chain applied to each input raw track is as follows: Dry Input → linear Gain → FIR EQ → DRC & Wet/Dry Mix → Distortion & Wet/Dry Mix → Pan → Reverb & Wet/Dry Mix → Sum with other stems. To drive each module, a set of parameters Θ_{module} are estimated. Refer to Figure 9.1 for a block diagram of the proposed system.

The gain, pan, EQ, and stereo reverb modules are parameterized and initialized identically to those in Sec 8.2, with one exception: the gain and pan modules are now initialized using the LSE method described in Sec 4.1. The DRC is parameterized and initialized according to Sec 7.3, and the distortion is parameterized and initialized according to Sec 6.3.

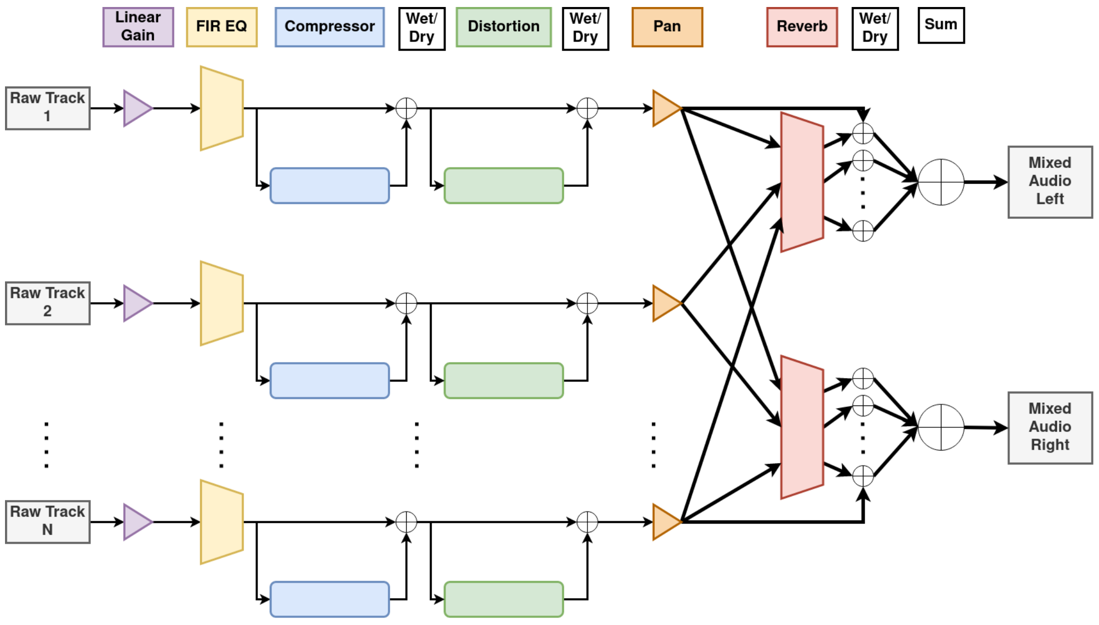


Figure 9.1: Signal chain of full reverse engineering system.

For each mixdown, four separate reverse engineering procedures are used that form an ablation study for the proposed system.

The first method utilises the full mixing chain in Fig 9.1. Once parameters are initialized the gradient descent with Adam optimizer and initial learning rate 10^{-4} first updates the gain, EQ, DRC, and panning parameters while bypassing the distortion and reverb modules for $40,000$ iterations or until early stopping is reached. Afterwards distortion is introduced to the mixing chain and jointly updated with gain, EQ, DRC, and panning parameters for another $40,000$ iterations or until early stopping is reached with learning rate 10^{-4}. Finally, reverb is introduced to the chain and all parameters are updated with a learning rate of 10^{-5} for another $40,000$ iterations or until early stopping is reached.

The second method (DRC-EQ-Reverb) excludes the use of the distortion module. To compensate, the gain, EQ, DRC, and panning parameters are allowed to update for $80,000$ iterations with learning rate 10^{-4} or until early stopping is reached. The reverb module is then introduced, the learning rate dropped to 10^{-5}, and the gradient descent proceeds for $40,000$ or until early stopping is reached.

The third method (EQ-Reverb) excludes both distortion and DRC, thus producing a linear mixdown similar to that of Sec 8.2. The gain, EQ, and pan parameters are allowed to update for $80,000$ iterations with learning rate 10^{-4} or until early stopping is reached. Afterwards the reverb module is introduced, the learning rate dropped to 10^{-5}, and descent proceeds for another $40,000$ iterations or until early stopping is reached.

The final method (Gain Mix), which acts as the baseline for the study, uses only gain and panning. This mix is calculated using the MSE method presented in Sec 4.1.

The reverse engineered mixdowns are measured against the reference mixdowns with an L1 MSS loss of resolutions 32768, 2048, 512, and 128 samples, which corresponds to windows of length 750ms, 50ms, 12ms, and 3ms.

9.3 Data

The multitracks used to evaluate the system were taken from the Cambridge Multitracks dataset [Senior, 2011]. Three multitracks performed by the alt-rock band Woodfire for their Weird Fear EP were chosen because the multitracks were all recorded in the same studio, by the same engineer, under similar circumstances. The song "Animals" was recorded to 15 tracks, the song "Haunted House" to 14 tracks, and the song "Wealthy in Time" to 13 tracks.

These time-aligned multitracks were given to student mix engineers at Queen Mary University of London to mix as part of their coursework, with each student responsible for producing one mixdown of the multitrack they were assigned. Students were given two hours to produce a mixdown of a song's first verse and chorus using the DAW of their choice and any plugins or automation they saw fit. Students were instructed to not alter the composition of any of the multitracks, but were allowed to mute any elements in the mix. Eight students produced mixdowns of "Animals," another eight students produced mixdowns of "Haunted House," and seven students produced mixdowns of "Wealthy in Time."

Ten seconds from each song's chorus was chosen for the reverse engineering task. Thus the system was tested against 23 mixdowns, ten seconds in length, across three multitracks. Ultimately 92 mixdowns were reverse engineered.

9.4 Results

9.4.1 Objective Results

The final MSS loss caluclated for each reverse engineered mixdown can be found in Table 9.1. Mixes XX_A are mixes of the song "Animals," XX_B are mixes of the song "Haunted House," and XX_C are mixes of "Wealthy in Time." Across all mixdowns, the full system performed best and the gain mix performed worst. 65% of the DRC-EQ-Reverb mixes outperformed the EQ-Reverb mixes on this objective measure. For the song "Animals," the full system performed best on the "03_A" and "08_A" mixdowns; for the song "Haunted House," the full system performed best on the "06_B" and "14_B" mixdowns; for the song "Wealthy in Time," the full system performed best on the "19_C" and "20_C" mixdowns.

Table 9.1: Multiscale spectrogram losses for reverse engineered mixes. Rows in bold denote the top two performing reverse engineered mixes using the full system for each song.

Mix	Full system	DRC-EQ-Reverb	EQ-Reverb	Gain Mix
00_A	1.2793	1.2840	1.2864	1.8987
01_A	1.8406	1.8683	1.9056	5.6233
02_A	1.5722	1.6723	1.7533	3.7127
03_A	**0.6690**	**0.7307**	**0.7295**	**1.3044**
04_A	1.5292	1.5734	1.5596	2.1578
05_A	1.1569	1.1815	1.2052	2.0023
08_A	**0.9358**	**0.9818**	**0.9945**	**1.5779**
17_A	1.9104	1.9148	1.9086	4.1754
06_B	**1.1986**	**1.3081**	**1.2858**	**2.8000**
07_B	2.7731	2.8395	2.8708	7.1260
09_B	1.9183	1.9733	1.9644	3.7378
10_B	5.9941	6.153	6.3025	11.0797
11_B	5.5800	5.9868	6.6175	9.5553
13_B	1.8684	2.0345	2.2830	4.2610
14_B	**1.5434**	**1.5575**	**1.5564**	**3.1416**
15_B	5.0719	5.2030	5.2575	8.6782
16_C	1.0712	1.0871	1.0842	1.8942
18_C	0.7481	0.8480	0.9614	2.1836
19_C	**0.6245**	**0.6469**	**0.6564**	**1.5719**
20_C	**0.5101**	**0.6606**	**0.6784**	**1.6957**
21_C	1.0388	1.2630	1.2925	2.4537
22_C	2.0082	2.0421	2.0868	4.5832
23_C	1.2846	1.3053	1.3028	2.0972

9.4.2 Perceptual Results

The top two performing reverse engineered mixes of each multitrack were chosen for a listener evaluation using the webMUSHRA framework [Schoeffler et al., 2018]. These mixdowns can be listened to at `https://jtcolonel.github.io/NonlinRevEng/`. All reference and reverse engineered mixdowns were normalized to -24.0 LUFS-I for the listening test.

Participants were presented with a reference mix and five stimuli, the four reverse engineered mixdowns plus a hidden reference, across six reference mixes. Participants were asked to rate each stimuli according to how closely it matched the reference, with 0 representing a poor match and 100 representing a perfect match. Participants were presented with the following preamble before taking the test:

> You will be provided with a reference mixdown at the top of the page. The task is then to rate the stimuli below based on how closely they match the reference. When evaluating how close two mixdowns are, one should consider how the individual elements of the multitrack are balanced in each of the mixdowns. This balance may include how loud elements are compared to one another, how these elements are spread in the stereo field, the tone of each element, the dynamic characteristics of each element, and how the mixdown coheres as a whole. Adjust the sliders for each example to rate the closeness, and use the whole scale when possible. A perfect score should constitute a mixdown that exactly matches the reference.

A total of 8 participants took part in the study, with an average age of 35 years and standard deviation of 7.08. 5 participants identified as men, and 3 as women. 7 participants reported having at least 4 years of experience with music production or audio engineering, and 1 participant reported no experience. No participants reported any diagnosed hearing impairments. Box and whisker plots of the participants' ratings are presented in Figure 9.2.

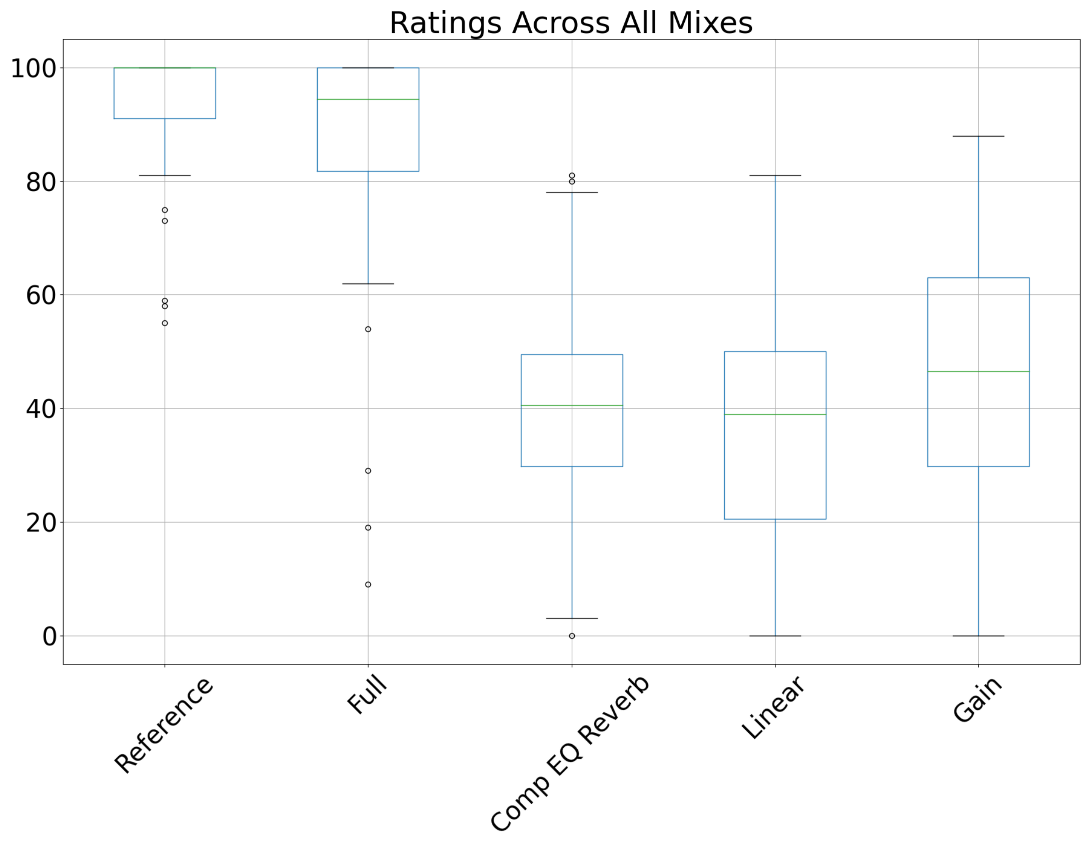


Figure 9.2: Box and whisker plots of the median, standard deviation and 95% confidence intervals of all participant ratings across all mixes. The listening test found that the mixdowns reverse engineered using the whole mixing chain were indistinguishable from a reference.

The analysis that follows is adapted from the perceptual study presented in Moffat and Reiss [2018]. The null hypothesis is that the perceptual evaluation scores are from the same distribution. A one-way ANOVA, with Bonferroni correction, shows for all stimuli that the effect each reverse engineering method had on user perception was statistically significant.

With the null hypothesis rejected, a post-hoc Tukey pairwise comparison, with Bonferroni correction to reduce the chance of type I errors, was used. Table 9.2 shows the results of these pairwise comparisons for all architectures used.

The pairwise comparisons demonstrate that across the six selected mixdowns, the perception of the full system's reverse engineered mixes do not differ significantly from the reference, and those of all other reverse engineering methods do differ significantly from the reference. These results hold for all but two mixdowns: 06_B and 19_C. In the case of 06_B, the perception of all reverse engineered mixes differ significantly from the reference. In the case of 19_C, the perception of both the full system and gain system's reverse engineered mixes do not differ significantly from the reference.

Table 9.2: Results of pairwise comparison of mixdown architecture on perceptual similarity rating across multitracks, with Bonferroni Correction, o $>$ 0.05, *<0.001 ·=no comparison. The listening test found that the mixdowns reverse engineered using the whole mixing chain were indistinguishable from a reference.

	Reference Mix	Full Mix	DRC-EQ-Rev Mix	Linear Mix	Gain Mix
Reference Mix	·	o	*	*	*
Full Mix	o	·	*	*	*
DRC-EQ-Rev Mix	*	*	·	o	o
Linear Mix	*	*	o	·	o
Gain Mix	*	*	o	o	·

9.5 Discussion

The results of both the objective evaluation and the listening test suggests that the full reverse engineering mixing chain outperforms all other mixing chains.

As mentioned in Sec 3.3.2, there are no rules of thumb when interpreting MSS loss values. However, some trends do appear within the objective evaluation. For example, within each mixdown the gain mix achieves the highest MSS loss, suggesting that just the introduction of EQ and reverb improves the system's matching performance. The DRC-EQ-Reverb mix slightly outperforms the EQ-Reverb mix on MSS loss, but not enough to make a convincing case that one is better than the other. It is interesting to note that all of the best performing mixdowns mentioned above achieve the lowest MSS for their gain mix approximations as well. This suggests that the full reverse engineering signal chain will perform best when the gain approximation is strong. However, in terms of perceptual evaluation the gain mix had a higher average rating than both the DRC-EQ-Reverb mix and linear mix.

The listening test results for mix 06_B demonstrate that the MSS loss measure does not necessarily measure perceptual closeness – even though the full system's reverse engineered 14_B mix had a greater MSS cost than that for 06_B, listener's rated the 14_B closer to its reference than 06_B. This contributes to a larger discussion in the literature regarding the need for more perceptually relevant cost functions for use in audio tasks [Manocha et al., 2021, Vahidi et al., 2023].

As well as providing mixdowns, students were asked to comment on the mixes of their peers. These comments provide potential explanations for what type of mixes the system performed poorly on. For example, across 16 comments on mix 10_B the word "creative" appears in five. One evaluation mentions that the "singer seems like he is on a completely different stage from the band," and another mentions that "the reverb/delay on the vocal was not quite fit." This suggests that the reverb and delay used on the vocal is probably distinct than that used on the rest of the mix, which the full mixing chain is not equipped to handle. This may explain why mix 10_B performed worst on the MSS objective measure.

For the reverse engineered mixes that do reach perceptual tolerance, the stems of the mixes can be bounced individually and compared with the students' comments. For example, 11 of 17 students mention that the vocals in the mix are too low when commenting on mix 08_A. After bouncing the stems using the mixing chain learned by the full system, the vocals measure -29.1 LUFS-I, compared to the full mix minus vocals which measures -23.8 LUFS-I. Pestana [2013] found that listeners prefer when vocals sit between -2 and 0 LU compared to the rest of the mix, which may explain why comments on the mix noted that the vocals were low.

For mix 20_C, several students commented that the kick drum is boomy and too prominent. The kick stem measures -25.6 LUFS-I, and the rest of the mix measures -26.0 LUFS-I. Pestana [2013] found that the main element of the mix ought to be within -2 to 0 LU of the rest of the mix, so the relative loudness of the kick suggests that it will draw focus. When observing the reverse engineered mix's individual effects as seen in Figure 9.3, the EQ on the kick has a slight boost between 60Hz and 300Hz. In terms of spectrum, the descriptor "boomy" is typically applied to elements within the range of 20-250 Hz [Man and Reiss, 2015], with Owsinski [2014] specifically suggesting the range 60-250 Hz. Figure 9.4 shows log frequency spectrograms of the raw kick track and processed kick stem. Here it can be seen that the reverse engineered effects processing increases the density of spectral energy below 200Hz, which may explain the "boomy" comments.

There are several directions future work can take. One is modelling automation, which is frequently used by mixing engineers. This could be realised using frame-by-frame approximations of mixing parameters, or some other control scheme. Another direction would be a method for learning what mixing chain may best suit a mix, rather than fixing the chain shown in Fig 9.1. This could entail identifying which effects have been applied to the raw tracks in the mixdown, similar to the work of Koo et al. [2022]. It would also be interesting to incorporate the time alignment and pitch correction shown in Schwarz and Fourer [2021], rather than forcing time and pitch alignment before applying the reverse engineering algorithm

This reverse engineering work may also aid in numerically characterizing mixing engineers' behaviour by analyzing and extracting mix parameters from a corpus of professional mixes. This corpus could then be used to improve objective measures of multitrack mixes for perceptual correlation to avoid issues such as those encountered when objectively measuring mixdowns.

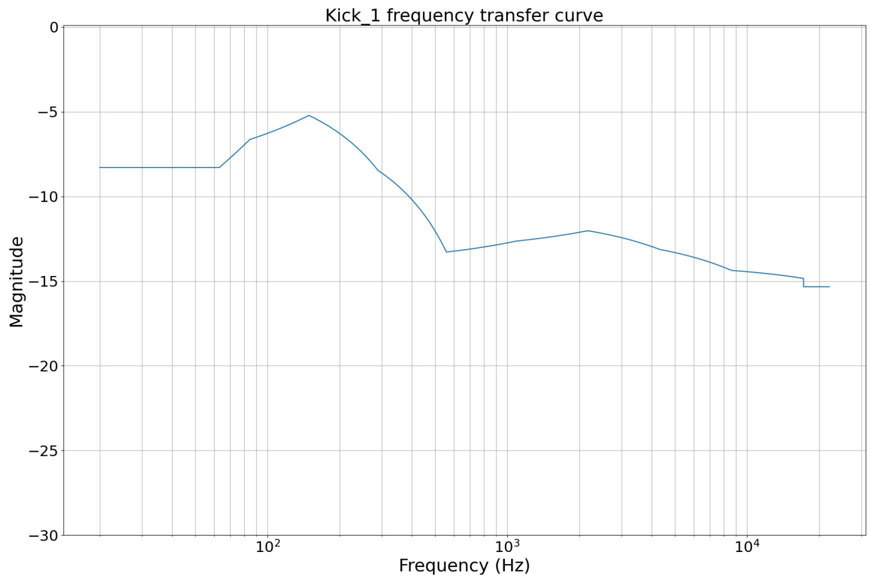


Figure 9.3: Learned EQ of kick drum raw track in mix 20_C. The slight boost in frequencies between 60Hz and 300Hz is associated with "boominess."

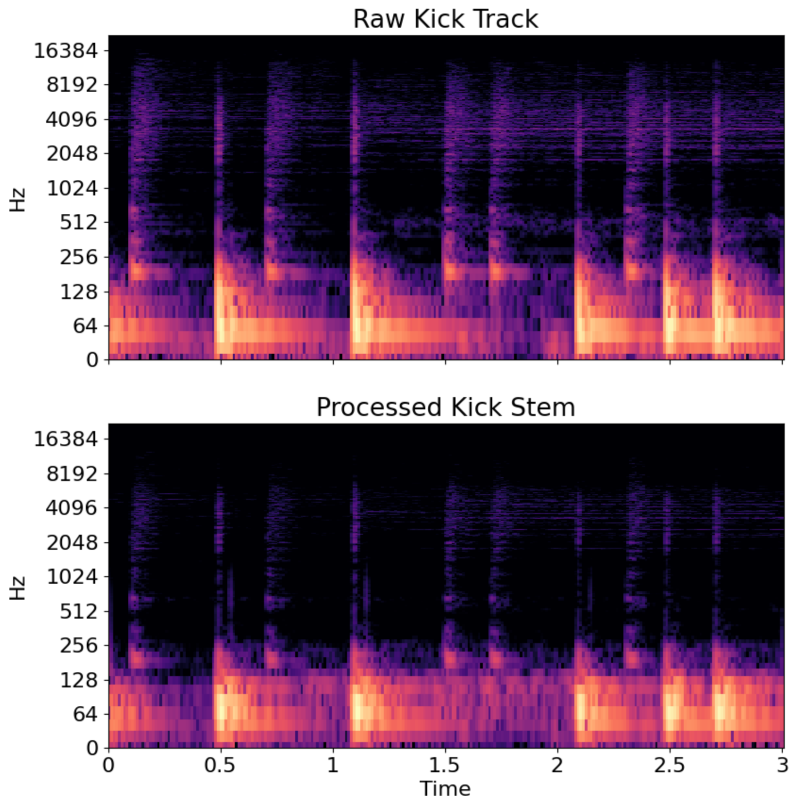


Figure 9.4: Log frequency spectrogram of the raw kick track and processed kick stem for mix 20_C. This demonstrates the reverse engineered effects removed most content above 250Hz and increased spectral density below 250Hz.

Chapter 10

Conclusions and further work

10.1 Summary of contributions

Throughout this dissertation, the paradigm of differentiable digital signal processing has been applied to audio effect modelling, culminating in a new method for reverse engineering a nonlinear mix of a multitrack recording.

The dissertation begins with demonstrations of FIR linear time-invariant effect matching in Chapter 4. Examples of gain, panning, and FIR EQ matching in the time domain were presented from previously published literature. Additionally, a method for matching a reverb IR given a dry/wet pair using gradient descent was described and compared to a reverb matching method using division in the frequency domain.

In Chapter 5, a novel neural network-based method for matching the coefficients of a cascade of biquads to an arbitrary magnitude response, named IIRNet, was proposed. This MLP learned to map a magnitude response to the parameters of a cascade of biquads via training on a dataset of filters whose coefficients are generated by random polynomials. An experiment demonstrated that training IIRNet on a combination of different families of random polynomials performed better than training on any individual family. IIRnet's accuracy and inference time were compared against two classical methods for IIR EQ matching: the modified Yule-Walker algorithm; and a gradient descent method. These methods were compared to one another using three datasets of filters: random filters generated using random polynomials; a dataset of guitar cabinet impulse responses; and a dataset of HRTFs. IIRNet was shown to be faster than the two traditional methods and most accurate across the three datasets.

Presented in Chapter 6 are methods for matching memoryless distortion effects. A differentiable Wiener-Hammerstein based model was shown to match memoryless waveshaping distortion effects. This model learned a cascade of an FIR pre-emphasis filter, gain parameter, waveshaper with DC removal, FIR de-emphasis filter, and volume parameter. Two novel families of parameterized waveshaping functions based on the hyperbolic tangent function, named the "SumTanh" and "PowTanh" families, were shown to outperform Fourier series-based and Legendre polynomial-based parameterized waveshapers as well as a baseline tanh waveshaper on objective measures. The PowTanh model performed best at matching a memoryless distortion effect when measured using a MSS loss. A listening test was also conducted to measure how well each method could match a reference perceptually. In the listening test, the SumTanh model performed best. Methods for DC removal were outlined for each of these waveshapers as well.

A method for matching dynamic range compression effects is shown in Chapter 7. A differentiable feedforward DRC was proposed that learns values for threshold, compression ratio, knee width, attack time, release time, smoothing time (to smooth attenuation while the DRC operates after attack and before release), and makeup gain. This DRC approximates attack and release time using approximate moving average filters. This approach avoids the sample-by-sample recursive calculations used in traditional digital DRCs, which when implemented differentiably require large amounts of time and memory to learn effects. The performance of this differentiable DRC was objectively measured on two dry/wet pairs of audio: one test signal proposed by Bitzer et al. [2006] that profiles the ballistics of a DRC; and one speech signal. Ultimately it was demonstrated that the DDSP DRC can match the compression on the test signal well, while slightly underperforming on the speech signal.

Chapter 8 revisits the problem of reverse engineering a multitrack mix using only linear time-invariant effects as proposed in Barchiesi and Reiss [2010]. In this task, a set of mixing parameters θ are learned that can map a raw multitrack recording to a mixdown. Here, θ defines the parameters for a mixing chain composed of gain, panning, FIR EQ, and reverb. The FIR EQs were formulated to approximate a graphic EQ by learning attenuations at specified octave bands and linearly interpolating a frequency transfer curve between those values. The performance of two differentiable mixing chains were compared to a gain/panning mix baseline. One mixing chain, called the stereo bus model, learned distinct reverb IRs for the left and right channel of the mixdown as well as wet/dry parameters for each track in the mulititrack. The other mixing chain, called the individual bus model, learned left and right reverb IRs for each individual track in the multitrack. Ultimately, these reverse engineering algorithms were applied to a dataset of six mixdowns produced using only linear time-invariant effects. An objective evaluation of the reverse engineering methods was performed by comparing extracted MIR features from the reverse engineered mixdowns to those of the reference mix, and this objective evaluation showed that the stereo bus architecture outperformed all other reverse engineered mixes. Then, a formal listening test was done to evaluate whether any of the reverse engineered mixes were perceptually indistinguishable from a reference. The outcome of this perceptual evaluation showed that only the mixdowns reverse engineered using the stereo bus architecture were perceptually indistinguishable from a reference mix.

Chapter 9 concludes the dissertation with a method for reverse engineering a nonlinear mix. A full mixing chain with gain, FIR EQ, DRC, distortion, panning, and reverb was proposed, and an ablation study was performed to measure the impact adding distortion and DRC has to the reverse engineered mixdowns. The full mixing chain was compared to a mixing chain without distortion, a mixing chain without DRC and distortion, and a gain/panning mix. When measured objectively using MSS loss, the full mixing chain was shown to outperform all other mixing chains on a dataset of 23 student mixes. Furthermore, a formal listening test showed that the reverse engineered mixes produced by the full mixing chain were perceptually indistinguishable from reference mixes on a subset of six student mixes.

Returning to the questions presented in Sec 1.2:

How can recent advances in machine learning for audio improve the technique of reverse engineering a mix?

This thesis has demonstrated that the paradigm of differentiable digital signal processing can be used to improve the technique of reverse engineering a mix originally proposed in Barchiesi and Reiss [2010]. In Barchiesi and Reiss [2010], methods were shown to match gain, panning, EQ, and DRC. The new technique for reverse engineering a mix proposed in this thesis can now model gain, panning, EQ, memoryless distortion, DRC, and reverb.

Furthermore, this dissertation sought to match mixes within perceptual tolerance, a goal not explicitly stated by Barchiesi and Reiss [2010]. When evaluating their algorithms, the authors of Barchiesi and Reiss [2010] used only objective measures related to how closely reverse engineered filters matched a reference, or how closely a reverse engineered mixdown's waveform matched a reference. In this dissertation, listening tests were performed to assess how closely reverse engineered mixdowns matched their reference. For both linear time-invariant mixdowns and nonlinear mixdowns, listening tests demonstrated that the reverse engineered mixdowns which performed best on objective measures also matched their reference mixdowns within perceptual tolerance.

The expansion of methods presented in this thesis allowed for the algorithms to be baselined on more natural mixdowns than those shown in Barchiesi and Reiss [2010]. Whereas Barchiesi and Reiss [2010] baselined their algorithm on three mixdowns produced by the authors, this dissertation baselined a method for reverse engineering linear time-invariant mix using six mixdowns mixed by a professional mix engineer, and baselined a method for reverse engineering a nonlinear mix using 23 mixdowns mixed by students.

Can a new approach to reverse engineering a mix simultaneously optimize the linear and nonlinear processing of the mix?

It was demonstrated that DDSP allows for the simultaneous optimization of linear and nonlinear processing due to its ability to incorporate gradient descent into matching effect parameters. As detailed in Chapters 8 & 9, a a piecemeal approach performed best. Rather than simultaneously optimizing all parameters in the mixdown, it is best to begin the reverse engineering with the gain and panning approximation shown in Barchiesi and Reiss [2010]. Then, EQ and DRC values are optimized along with gain and panning. Afterwards, distortion is added to the optimization. Finally, reverberation is added and all parameters are optimized until early stopping is reached. This demonstrates a clear improvement to Barchiesi and Reiss [2010], which presented two separate algorithms: one for matching linear processing, and one for matching DRC.

Can a mix be reverse engineered in the absence of information about how the mixdown was produced?

In Chapter 9, few restrictions were placed on the types of processing the students could use to make their mix. Students were allowed to mix using the DAW of their choice, any plugins they wished to use, and could apply their effects in any order they saw fit. Furthermore, the methodology makes no requirements that all mixing happen within a DAW; it is theoretically possible to mix fully analog, or use a hybrid analog/digital mixing chain.

The method does assume that time alignment has already been performed (i.e. is not part of the mixing process), and that no pitch correction takes place in mixing. Furthermore, the method assumes that no automation has been applied to any parameters in the mixing chain.

Which audio effects need to be emulated in order to reverse engineer a mix within perceptual tolerance?

Gain, panning, EQ, reverb, distortion, and DRC were all required in order to match student mixes within perceptual tolerance. Mixing chains without all of these effects could not reverse engineer the student mixdonws within perceptual tolerance.

As a result, novel formulations for DDSP memoryless distortion, DRC, and IIR EQ matching were presented. IIRNet, shown in Chapter 5, represents an improvement to the magnitude response matching proposed in Nercessian [2020] and contributes to a larger body of literature regarding IIR filter estimation. In Chapter 6, the DDSP W-H method represents an improvement to the approach detailed in Nercessian et al. [2021], as the waveshaping action is much more interpretable, and the use of only two emphasis filters is easier to inspect and modify than the cascade of six filters and tanh nonlinearities of Nercessian et al. [2021]. Additionally, the DDSP DRC shown in Chapter 7 is able to separately model the attack and release action of a DRC without the use of recursive calculation, making it more lightweight than the method shown in Wright et al. [2022] and more expressive than the model shown in Steinmetz et al. [2022]. Ultimately, these modules contribute to an expanding body of DDSP literature.

Can the parameters of a reverse engineered mixdown be made legible to a mix engineer?

All of the learned parameters in the reverse engineered mixes presented in this thesis can be easily read by a mix engineer. The "graphic EQ" formulation used in Chapters 6, 8, & 9 presents a legible frequency transfer curve that ought to be familiar to anyone who has used a graphic EQ while mixing. The waveshaping module used in the distortion effect prints a clear waveshaping transfer curve, as demonstrated in Fig 6.4. The parameters learned by the DRC module follow a traditional implementation of a digital DRC, as shown in Table 7.1. Finally, the stereo bus reverb architecture explicitly learns two reverb IRs, as shown in Fig 8.8. While these reverb IRs may be difficult to modify, they are completely explained.

Previous examples of automatic multitrack mixing, like the algorithms presented in Martínez-Ramírez et al. [2022] or Martínez Ramírez et al. [2021], rely on blackbox neural network audio effects to mix multitracks. As such, they employ neural network operations that are unfamiliar to those outside the field. The work presented in this dissertation has provided the field of automatic mixing with greybox, differentiable DSP modules that are inspired by the effects used by mix engineers. Therefore, they represent an important step in bringing automatic mixing tools to practitioners and encouraging intervention in machine-made automatic mixes.

10.2 Further work

There are several opportunities to continue modelling various individual audio effects with differentiable greybox techniques. In terms of linear effects, a dedicated delay/echo module could improve the performance of the reverse engineered mixes. Currently, the reverb module is tasked with jointly learning reverberation across the entire mix as well as delay/echo effects that may have been applied to individual tracks in the multitrack.

Also, the inclusion of a parameterized differentiable reverb module would also be desirable. The proposed model learns the individual taps of a reverb IR, i.e. the reverb module currently learns 88200 parameters for a given mix. The inclusion of a differentiable feedback delay network, such as that in [Ibnyahya and Reiss, 2022], would reduce the number of learnable parameters as well as increasing the legibility and control of the learned effect.

There are several nonlinear effects that can also be implemented in a differentiable greybox manner. For example, the development of a differentiable distortion effect with memory would be desirable as many mix engineers use these effects in their process. In addition, a more accurate differentiable DRC may improve system performance. Additionally a differentiable noise gate effect has yet to be proposed in the literature, which is an effect commonly used by mix engineers.

This work did not attempt to model any automation in the mixing chain, which is quite restrictive when trying to model professional mix engineering behaviours. It may be possible to model automation in the following way. First, a mixing chain can be learned to match an excerpt of the target mixdown (i.e. the first second of the mix). Then, these learned parameters can be used to initialize a gradient descent performed on the next portion of the mix, and so on. Should no drastic automation take place between two excerpts, one would expect the parameters to only change slightly. Finally, a smoothing procedure such as a Savtizky-Golay filter can be used to interpolate these automations across the mix.

In addition, this work did not attempt to model any time-alignment or pitch correction, which are two techniques that are typically expected of a mix engineer. Inspiration can be taken from Schwarz and Fourer [2021], in which time-alignment and pitch correction are used to reverse engineer DJ mixes. It may be possible, for example, to use the two-tap delay filters mentioned in Nercessian et al. [2021] to perform a differentiable alignment. To improve methods for reverse engineering DJ mixes, the differentiable effects modules presented in this dissertation could be combined with the methods shown in Schwarz and Fourer [2021] to not only reverse engineer time alignment and pitch correction, but also the effects used to blend two tracks together (e.g. high-pass filtering, DRC, etc.).

Determining what effects may have been applied to individual tracks in the raw track is also an open question. The proposed methodology assumes a fixed mixing chain for each raw track, which may limit its modelling capabilities. Cues can be taken from Steinmetz et al. [2022], where an ANN audio effects encoder learns audio effects related information from a reference music recording, and from Koo et al. [2022], where a similar methodolgy was used to apply the style of one multitrack mix to a separate mixdown. Learning a unique mixing chain for each mixdown may improve the performance of the method as well as provide another paradigm for modelling individual mix engineer's practice.

Finally, a multimodal analysis that combines comments on mixdowns with the learned effects of the reverse engineered mixdowns may also deepen our understanding of mix engineering as a practice. Research ventures, such as the FAST IMPACt (Fusing Audio Semantic Technologies for Intelligent Music Production and Consumption) initiative [1], have published much literature that tries to tie semantic descriptions of mixes with objective features calculated on the mix. This reverse engineering technique can allow for a more fine-grained approach to numerically characterizing common mix descriptors. In the same vein, this new technique may help reduce the need for collection and maintenance of DAW recording sessions when studying mix engineering, which could increase the scope of ethnographic inquiries such as those shown in Pras et al. [2018], or studies of analog and digital mixing techniques like that in Chambers-Moranz et al. [2019].

[1]http://www.semanticaudio.ac.uk/publications-2/ Accessed 21 April, 2023

Bibliography

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

Jont Allen. Short term spectral analysis, synthesis, and modification by discrete fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 25(3):235–238, 1977. doi: 10.1109/TASSP.1977.1162950.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv:1607.06450*, 2016.

Daniele Barchiesi and Joshua Reiss. Reverse engineering of a mix. *Journal of the Audio Engineering Society*, 58(7/8):563–576, 2010.

Søren Bech and Nick Zacharov. *Perceptual audio evaluation-Theory, method and application.* John Wiley & Sons, 2007.

Albert T Bharucha-Reid and Masilamani Sambandham. *Random polynomials: Probability and mathematical statistics: a series of monographs and textbooks.* Academic Press, 2014.

Stefan Bilbao, Fabián Esqueda, Julian D. Parker, and Vesa Välimäki. Antiderivative antialiasing for memoryless nonlinearities. *IEEE Signal Processing Letters*, 24(7):1049–1053, 2017. doi: 10.1109/LSP.2017.2675541.

Chris M Bishop. Neural networks and their applications. *Review of scientific instruments*, 65 (6):1803–1832, 1994.

Joerg Bitzer, Denny Schmidt, and Uwe Simmer. Parameter estimation of dynamic range compressors: models, procedures and test signals. In *Audio Engineering Society Convention 120.* Audio Engineering Society, 2006.

Dmitry Bogdanov, Nicolas Wack, Emilia Gómez Gutiérrez, Sankalp Gulati, Herrera Boyer, Oscar Mayor, Gerard Roma Trepat, Justin Salamon, José Ricardo Zapata González, Xavier Serra, et al. Essentia: An audio analysis library for music information retrieval. In *Britto A, Gouyon F, Dixon S, editors. 14th Conference of the International Society for Music Information Retrieval (ISMIR); 2013 Nov 4-8; Curitiba, Brazil.[place unknown]: ISMIR; 2013. p. 493-8.* International Society for Music Information Retrieval (ISMIR), 2013.

Korneel van den Broek. Mp3net: coherent, minute-long music generation from raw audio with a simple convolutional gan. *arXiv preprint arXiv:2101.04785*, 2021.

Gary Bromham, Dave Moffat, Mathieu Barthet, and György Fazekas. The impact of compressor ballistics on the perceived style of music. In *Audio Engineering Society Convention 145*. Audio Engineering Society, 2018.

Gary Bromham, David Moffat, Di Sheng, and György Fazekas. Measuring audibility threshold levels for attack and release in a dynamic range compressor. *Journal of the Audio Engineering Society*, october 2022.

Recommendation ITU-R BS.1534-3. *Method for the subjective assessment of intermediate quality level of coding systems.* International Telecommunication Union, 2003.

Richard James Burgess. *The art of music production: The theory and practice.* Oxford University Press, 2013.

Richard James Burgess. *The history of music production.* Oxford University Press, 2014.

Anne Caclin, Stephen McAdams, Bennett K Smith, and Suzanne Winsberg. Acoustic correlates of timbre space dimensions: A confirmatory study using synthetic tones. *The Journal of the Acoustical Society of America*, 118(1):471–482, 2005.

Alex Case. *Mix smart: Pro audio tips for your multitrack mix.* Focal Press, 2011.

Franco Caspe, Andrew McPherson, and Mark Sandler. Ddx7: Differentiable fm synthesis of musical instrument sounds. *arXiv preprint arXiv:2208.06169*, 2022.

Rafael Cauduro Dias de Paiva, Jyri Pakarinen, Vesa Välimäki, and Miikka Tikander. Real-time audio transformer emulation for virtual tube amplifiers. *EURASIP Journal on Advances in Signal Processing*, 2011:1–15, 2011.

Rafael Cauduro Dias de Paiva, Jyri Pakarinen, and Vesa Välimäki. Reduced-complexity modeling of high-order nonlinear audio systems using swept-sine and principal component analysis. In *Audio Engineering Society Conference: 45th International Conference: Applications of Time-Frequency Processing in Audio*. Audio Engineering Society, 2012.

Ryland Chambers-Moranz, Amandine Pras, and Nate Thomas. The generation gap—perception and workflow of analog vs. digital mixing. In *Audio Engineering Society Convention 147*. Audio Engineering Society, 2019.

Yiu-Tong Chan and Randy Langford. Spectral estimation via the high-order yule-walker equations. *IEEE Trans. Acoust.*, 30(5), 1982.

Michael Chemistruck, Kyle Marcolini, and Will Pirkle. Generating matrix coefficients for feedback delay networks using genetic algorithm. In *Audio Engineering Society Convention 133*. Audio Engineering Society, 2012.

Bo-Yu Chen, Wei-Han Hsu, Wei-Hsiang Liao, Marco A Martínez Ramírez, Yuki Mitsufuji, and Yi-Hsuan Yang. Automatic dj transitions with differentiable audio effects and generative adversarial networks. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 466–470. IEEE, 2022.

Taishih Chi, Powen Ru, and Shihab A Shamma. Multiresolution spectrotemporal analysis of complex sounds. *The Journal of the Acoustical Society of America*, 118(2):887–906, 2005.

Miguel Civit, Javier Civit-Masot, Francisco Cuadrado, and Maria J Escalona. A systematic review of artificial intelligence-based music generation: Scope, applications, and future trends. *Expert Systems with Applications*, page 118190, 2022.

Jay Coggin and Will Pirkle. Automatic design of feedback delay network reverb parameters for impulse response matching. In *Audio Engineering Society Convention 141*. Audio Engineering Society, 2016.

Joseph Colonel and Joshua D Reiss. Exploring preference for multitrack mixes using statistical analysis of mir and textual features. In *Audio Engineering Society Convention 147*. Audio Engineering Society, 2019.

Joseph T Colonel and Joshua Reiss. Reverse engineering of a recording mix with differentiable digital signal processing. *The Journal of the Acoustical Society of America*, 150(1):608–619, 2021.

Marco Comunità, Dan Stowell, and Joshua D. Reiss. Guitar effects recognition and parameter estimation with convolutional neural networks. *J. Audio Eng. Soc*, 69(7/8):594–604, 2021. URL `http://www.aes.org/e-lib/browse.cfm?elib=21124`.

Stefano D'Angelo, Jyri Pakarinen, and Vesa Valimaki. New family of wave-digital triode models. *IEEE transactions on audio, speech, and language processing*, 21(2):313–321, 2012.

Christian Darken, Joseph Chang, John Moody, et al. Learning rate schedules for faster stochastic gradient search. In *Neural networks for signal processing*, volume 2. Citeseer, 1992.

Brecht De Man. *Towards a better understanding of mix engineering*. PhD thesis, Queen Mary University of London, 2017.

Brecht De Man and Joshua D Reiss. A knowledge-engineered autonomous mixing system. In *Audio Engineering Society Convention 135*. Audio Engineering Society, 2013.

Brecht De Man and Joshua D Reiss. Adaptive control of amplitude distortion effects. In *Audio engineering society conference: 53rd international conference: Semantic audio*. Audio Engineering Society, 2014.

Brecht De Man, Ryan Stables, and Joshua D Reiss. *Intelligent Music Production*. Routledge, 2019.

Giovanni De Sanctis and Augusto Sarti. Virtual analog modeling in the wave-digital domain. *IEEE transactions on audio, speech, and language processing*, 18(4):715–727, 2009.

Monica Dinculescu, Jesse Engel, and Adam Roberts. Midime: Personalizing a musicvae model with user data.

Peter Dodds. A flexible numerical optimization approach to the design of biquad filter cascades. In *149th AES Convention*, 2020.

Richard C Dorf and Robert H Bishop. *Modern control systems*. Pearson, 2011.

J Stephen Downie. Music information retrieval. *Annual review of information science and technology*, 37(1):295–340, 2003.

W Ross Dunkel, Maximilian Rest, Kurt James Werner, Michael Jørgen Olsen, and Julius O Smith III. The fender bassman 5f6-a family of preamplifier circuits—a wave digital filter case study. In *Proceedings of the 19th International Conference on Digital Audio Effects (DAFx-16), Brno, Czech Republic*, pages 5–9, 2016.

Stefano D'Angelo and Vesa Välimäki. Generalized moog ladder filter: Part ii–explicit nonlinear model through a novel delay-free loop implementation method. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(12):1873–1883, 2014.

Douglas Eck and Juergen Schmidhuber. A first look at music composition using lstm recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, 103(4):48, 2002.

Alan Edelman, Eric Kostlan, and Michael Shub. How many eigenvalues of a random matrix are real? *J. Amer. Math. Soc.*, 7(1), 1994. ISSN 0894-0347. doi: 10.2307/2152729.

Felix Eichas and Udo Zölzer. Black-box modeling of distortion circuits with block-oriented models. In *Proceedings of the International Conference on Digital Audio Effects (DAFx), Brno, Czech Republic*, pages 5–9, 2016.

Felix Eichas and Udo Zölzer. Gray-box modeling of guitar amplifiers. *Journal of the Audio Engineering Society*, 66(12):1006–1015, 2018.

Felix Eichas, Stephan Möller, and Udo Zölzer. Block-oriented modeling of distortion audio effects using iterative minimization. *Proc. Digital Audio Effects (DAFx-15), Trondheim, Norway*, 2015.

Felix Eichas, Stephan Möller, and Udo Zölzer. Block-oriented gray box modeling of guitar amplifiers. In *Proceedings of the International Conference on Digital Audio Effects (DAFx), Edinburgh, UK*, pages 5–9, 2017.

Taffeta M Elliott, Liberty S Hamilton, and Frédéric E Theunissen. Acoustic structure of the five perceptual dimensions of timbre in orchestral instrument tones. *The Journal of the Acoustical Society of America*, 133(1):389–404, 2013.

Jesse Engel, Chenjie Gu, Adam Roberts, et al. Ddsp: Differentiable digital signal processing. In *International Conference on Learning Representations*, 2019.

Fabián Esqueda, Henri Pöntynen, Julian D Parker, and Stefan Bilbao. Virtual analog models of the lockhart and serge wavefolders. *Applied Sciences*, 7(12):1328, 2017.

Fabián Esqueda, Boris Kuznetsov, and Julian D. Parker. Differentiable white-box virtual analog modeling. In *2021 24th International Conference on Digital Audio Effects (DAFx)*, pages 41–48, 2021.

Antoine Falaize and Thomas Hélie. Passive guaranteed simulation of analog audio circuits: A port-hamiltonian approach. *Applied Sciences*, 6(10):273, 2016.

Michel Foucoult. Discipline and punish. *A. Sheridan, Tr., Paris, FR, Gallimard*, 1975.

Thomas Funkhouser, Ingrid Carlbom, Gary Elko, Gopal Pingali, Mohan Sondhi, and Jim West. A beam tracing approach to acoustic modeling for interactive virtual environments. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 21–32, 1998.

Dimitrios Giannoulis, Michael Massberg, and Joshua D Reiss. Digital dynamic range compressor design—a tutorial and analysis. *Journal of the Audio Engineering Society*, 60(6): 399–408, 2012.

Dimitrios Giannoulis, Michael Massberg, and Joshua D Reiss. Parameter automation in a dynamic range compressor. *Journal of the Audio Engineering Society*, 61(10):716–726, 2013.

Olivier Gillet and Gaël Richard. Enst-drums: an extensive audio-visual database for drum signals processing. In *ISMIR*, pages 156–159, 2006.

Jean Ginibre. Statistical ensembles of complex, quaternion, and real matrices. *J. Mathematical Phys.*, 6, 1965. ISSN 0022-2488. doi: 10.1063/1.1704292.

Stanislaw Gorlow and Sylvain Marchand. Reverse engineering stereo music recordings pursuing an informed two-stage approach. In *2013 International Conference on Digital Audio Effects (DAFx-13)*, pages 1–8, 2013.

James Douglas Hamilton. *Time series analysis.* Princeton University Press, 1994.

John M. Hammersley. The zeros of a random polynomial. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, 1954–1955, vol. II*, 1956.

Scott Hawley, Benjamin Colburn, and Stylianos Ioannis Mimilakis. Profiling audio compressors with deep neural networks. In *Audio Engineering Society Convention 147*. Audio Engineering Society, 2019.

Ben Hayes, Charalampos Saitis, and György Fazekas. Neural waveshaping synthesis. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

Thomas Hélie. On the use of volterra series for real-time simulations of weakly nonlinear analog audio devices: Application to the moog ladder filter. In *Proceedings of the 9th International Conference on Digital Audio Effects (DAFx'06)*, pages 7–12. Citeseer, 2006.

Russ Hepworth-Sawyer and Jay Hodgson. *Mixing music.* Taylor & Francis, 2016.

Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. Cnn architectures for large-scale audio classification. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 131–135. IEEE, 2017.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

Martin Holters and Udo Zölzer. A generalized method for the derivation of non-linear state-space models from circuit schematics. In *2015 23rd European Signal Processing Conference (EUSIPCO)*, pages 1073–1077. IEEE, 2015.

R. Huber and B. Kollmeier. Pemo-q—a new method for objective audio quality assessment using a model of auditory perception. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(6):1902–1911, 2006. doi: 10.1109/TASL.2006.883259.

Ilias Ibnyahya and Joshua D Reiss. A method for matching room impulse responses with feedback delay networks. In *Audio Engineering Society Convention 153*. Audio Engineering Society, 2022.

Il'dar A. Ibragimov and Nina B. Maslova. The mean number of real zeros of random polynomials. I. Coefficients with zero mean. *Teor. Verojatnost. i Primenen.*, 16, 1971. ISSN 0040-361x.

Roey Izhaki. *Mixing audio: concepts, practices, and tools.* Focal Press, 2008.

Nicholas Jillings, Brecht De Man, David Moffat, Joshua D Reiss, et al. Web audio evaluation tool: A browser-based listening test environment. 2015.

Zakhar Kabluchko. Critical points of random polynomials with independent identically distributed roots. *Proc. Amer. Math. Soc.*, 143(2), 2015. ISSN 0002-9939. doi: 10.1090/S0002-9939-2014-12258-1.

Mark Kac. On the average number of real roots of a random algebraic equation. *Bull. Amer. Math. Soc.*, 49, 1943. ISSN 0002-9904. doi: 10.1090/S0002-9904-1943-07912-8.

Christoph Kemper. Musical instrument with acoustic transducer, August 5 2014. US Patent 8,796,530.

Gary S. Kendall. The decorrelation of audio signals and its impact on spatial imagery. *Computer Music Journal*, 19(4):71–87, 1995. ISSN 01489267, 15315169. URL `http://www.jstor.org/stable/3680992`.

Adrian Kim, Soram Park, Jangyeon Park, Jung-Woo Ha, Taegyun Kwon, and Juhan Nam. Automatic dj mix generation using highlight detection. *Proc. ISMIR, late-breaking demo paper*, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398, 2021.

Takao Kobayashi and Satoshi Imai. Design of IIR digital filters with arbitrary log magnitude function by WLS techniques. *IEEE Trans. Acoust.*, 1990.

Bennett Kolasinski. A framework for automatic mixing using timbral similarity measures and genetic optimization. *journal of the audio engineering society*, may 2008.

Junghyun Koo, Marco A Martinez-Ramirez, Wei-Hsiang Liao, Stefan Uhlich, Kyogu Lee, and Yuki Mitsufuji. Music mixing style transfer: A contrastive learning approach to disentangle audio effects. *arXiv preprint arXiv:2211.02247*, 2022.

Abhijit Kulkarni, SK Isabelle, and HS Colburn. Sensitivity of human subjects to head-related transfer-function phase spectra. *JASA*, (5), 1999.

Boris Kuznetsov, Julian D Parker, and Fabián Esqueda. Differentiable iir filters for machine learning applications. In *Proc. Int. Conf. Digital Audio Effects (eDAFx-20)*, pages 297–303, 2020.

Mathias C Lang. Weighted least squares IIR filter design with arbitrary magnitude and phase responses and specified stability margin. In *IEEE Symp. on Adv. in Dig. Filt. and Sig. Proc.*, 1998.

Olivier Lartillot, Petri Toiviainen, and Tuomas Eerola. A matlab toolbox for music information retrieval. In *Data analysis, machine learning and applications*, pages 261–268. Springer, 2008.

Sungho Lee, Hyeong-Seok Choi, and Kyogu Lee. Differentiable artificial reverberation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2022.

Heiner Löllmann, Emre Yilmaz, Marco Jeub, and Peter Vary. An improved algorithm for blind reverberation time estimation. In *Proceedings of international workshop on acoustic echo and noise control (IWAENC)*, pages 1–4, 2010.

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017. URL `http://arxiv.org/abs/1711.05101`.

Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*, 2019.

Jianwen Luo, Kui Ying, and Jing Bai. Savitzky–golay smoothing and differentiation filter for even number data. *Signal processing*, 85(7), 2005.

Zheng Ma, Brecht De Man, Pedro DL Pestana, Dawn AA Black, and Joshua D Reiss. Intelligent multitrack dynamic range compression. *Journal of the Audio Engineering Society*, 63(6): 412–426, 2015.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

Jaromír Mačák. *Real-time digital simulation of guitar amplifiers as audio effects.* PhD thesis, Ph. D. thesis, Brno University of Technology, Brno, 2012.

Shoji Makino. *Audio source separation*, volume 433. Springer, 2018.

BD Man and Joshua D Reiss. Analysis of peer reviews in music production. *Journal on the Art of Record Production*, 2015.

Pranay Manocha, Zeyu Jin, Richard Zhang, and Adam Finkelstein. CDPAM: Contrastive learning for perceptual audio similarity. In *ICASSP 2021, To Appear*, June 2021.

Marco A Martínez Ramírez. *Deep learning for audio effects modeling.* PhD thesis, Queen Mary University of London, 2021.

Marco A Martínez-Ramírez, Wei-Hsiang Liao, Giorgio Fabbro, Stefan Uhlich, Chihiro Nagashima, and Yuki Mitsufuji. Automatic music mixing with deep learning and out-of-domain data. *arXiv preprint arXiv:2208.11428*, 2022.

Marco A. Martínez Ramírez, Emmanouil Benetos, and Joshua D. Reiss. Modeling plate and spring reverberation using a dsp-informed deep neural network. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 241–245, 2020. doi: 10.1109/ICASSP40776.2020.9053093.

Marco A Martínez Ramírez, Daniel Stoller, and David Moffat. a deep learning approach to intelligent drum mixing with the wave-u-net. *Journal of the Audio Engineering Society*, 69 (3):142–151, march 2021. doi: https://doi.org/10.17743/jaes.2020.0031.

Brian McFee, Colin Raffel, Dawen Liang, Daniel P Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, pages 18–25. Citeseer, 2015.

Madan L. Mehta. *Random matrices and the statistical theory of energy levels.* Academic Press, New York-London, 1967.

Marcus Michelen. Real roots near the unit circle of random polynomials. *Trans. Amer. Math. Soc.*, 2021.

Marcus Michelen and Julian Sahasrabudhe. Random polynomials: the closest roots to the unit circle. *arXiv:2010.10869*, 2020.

George A Miller. The masking of speech. *Psychological bulletin*, 44(2):105, 1947.

Tom M Mitchell and Tom M Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.

David Moffat and Joshua D Reiss. Perceptual evaluation of synthesized sound effects. *ACM Transactions on Applied Perception (TAP)*, 15(2):1–19, 2018.

David Moffat, Brecht De Man, and Joshua D Reiss. Semantic music production: A meta-study. *Journal of the Audio Engineering Society*, 70(7/8):548–564, 2022.

Masanori Morise, Fumiya Yokomori, and Kenji Ozawa. World: a vocoder-based high-quality speech synthesis system for real-time applications. *IEICE TRANSACTIONS on Information and Systems*, 99(7):1877–1884, 2016.

Michael C Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science*, 6(2-3):247–280, 1994.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

Ervk Najduchowski, Marcin Lewandowski, and Piotr Bobinski. Automatic audio mastering system. In *2018 Joint Conference - Acoustics*, pages 1–6, 2018. doi: 10.1109/ACOUSTICS. 2018.8502427.

Shahan Nercessian. Neural parametric equalizer matching using differentiable biquads. 2020.

Shahan Nercessian. End-to-end zero-shot voice conversion using a ddsp vocoder. In *2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 1–5, 2021. doi: 10.1109/WASPAA52581.2021.9632754.

Shahan Nercessian. Differentiable world synthesizer-based neural vocoder with application to end-to-end audio style transfer. *arXiv preprint arXiv:2208.07282*, 2022.

Shahan Nercessian, Andy Sarroff, and Kurt James Werner. Lightweight and interpretable neural modeling of an audio distortion effect using hyperconditioned differentiable biquads. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 890–894. IEEE, 2021.

Arlene C Neuman, Matthew H Bakke, Carol Mackersie, Sharon Hellman, and Harry Levitt. The effect of compression ratio and release time on the categorical rating of sound quality. *The Journal of the Acoustical Society of America*, 103(5):2273–2281, 1998.

Antonin Novak, Laurent Simon, Pierrick Lotton, and Frantisek Kadlec. Modeling of nonlinear audio systems using swept-sine signals: Application to audio effects. In *Proc. of the 12th Int. Conference on Digital Audio Effects (DAFx-09)*, pages 1–4, 2009.

Antonin Novak, Laurent Simon, and Pierrick Lotton. Analysis, synthesis, and classification of nonlinear systems using synchronized swept-sine method for audio effects. *EURASIP Journal on Advances in Signal Processing*, 2010:1–8, 2010a.

Antonin Novak, Laurent Simon, Pierrick Lotton, and Joël Gilbert. Chebyshev model and synchronized swept sine method in nonlinear audio effect modeling. In *Proc. 13th Int. Conference on Digital Audio Effects (DAFx-10)*, page 15, 2010b.

Antonin Novak, Laurent Simon, and Pierrick Lotton. Extension of generalized hammerstein model to non-polynomial inputs. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pages 21–25. IEEE, 2016.

Takeshi Okuzono, Takumi Yoshida, and Kimihiro Sakagami. Efficiency of room acoustic simulations with time-domain fem including frequency-dependent absorbing boundary conditions: Comparison with frequency-domain fem. *Applied Acoustics*, 182:108212, 2021.

Alan V Oppenheim, John Buck, Michael Daniel, Alan S Willsky, Syed Hamid Nawab, and Andrew Singer. *Signals & systems.* Pearson Educación, 1997.

Simone Orcioni, Alessandro Terenzi, Stefania Cecchi, Francesco Piazza, and Alberto Carini. Identification of volterra models of tube audio devices using multiple-variance method. *Journal of the Audio Engineering Society*, 66(10):823–838, 2018.

Bobby Owsinski. *The mixing engineer's handbook.* Course Technology, Cengage Learning, 2014.

Julian D Parker, Vadim Zavalishin, and Efflam Le Bivic. Reducing the aliasing of nonlinear waveshaping using continuous-time convolution. In *Proc. Int. Conf. Digital Audio Effects (DAFx-16), Brno, Czech Republic*, pages 137–144, 2016.

Julian D Parker, Fabián Esqueda, and André Bergner. Modelling of nonlinear state-space systems using a deep neural network. In *Proceedings of the International Conference on Digital Audio Effects (DAFx), Birmingham, UK*, pages 2–6, 2019.

Julian D. Parker, Sebastian J. Schlecht, Rudolf Rabenstein, and Maximilian Schäfer. Physical modeling using recurrent neural networks with fast convolutional layers, 2022. URL `https://arxiv.org/abs/2204.10125`.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Geoffroy Peeters, Bruno L Giordano, Patrick Susini, Nicolas Misdariis, and Stephen McAdams. The timbre toolbox: Extracting audio descriptors from musical signals. *The Journal of the Acoustical Society of America*, 130(5):2902–2916, 2011.

Robin Pemantle and Igor Rivin. The distribution of zeros of the derivative of a random polynomial. In *Advances in combinatorics*. Springer, Heidelberg, 2013.

Giovanni Pepe, Leonardo Gabrielli, Stefano Squartini, and Luca Cattani. Designing audio equalization filters by deep neural networks. *Applied Sciences*, 2020.

Pedro Pestana. *Automatic Mixing Systems Using Adaptive Audio Effects*. PhD thesis, Universidade Catolica Portuguesa, 2013.

Amandine Pras, Brecht De Man, and Joshua D Reiss. A case study of cultural influences on mixing practices. In *Audio Engineering Society Convention 144*. Audio Engineering Society, 2018.

Amandine Pras, Kierian Turner, Toby Bol, and Emmanuelle Olivier. Production processes of pop music arrangers in bamako, mali. In *Audio Engineering Society Convention 147*. Audio Engineering Society, 2019.

Andrea Primavera, Stefania Cecchi, Laura Romoli, Michele Gasparini, and Francesco Piazza. Approximation of dynamic convolution exploiting principal component analysis: Objective and subjective quality evaluation. In *Audio Engineering Society Convention 133*. Audio Engineering Society, 2012a.

Andrea Primavera, Stefania Cecchi, Laura Romoli, Michele Gasparini, and Francesco Piazza. An efficient dsp implementation of a dynamic convolution approach using principal component analysis. In *2012 5th European DSP Education and Research Conference (EDERC)*, pages 30–34, 2012b. doi: 10.1109/EDERC.2012.6532219.

Lawrence Rabiner, Nancy Graham, and Howard Helms. Linear programming design of IIR digital filters with arbitrary magnitude function. *IEEE Trans. Acoust.*, 1974.

Lawrence R Rabiner and Bernard Gold. *Theory and application of digital signal processing.* Englewood Cliffs: Prentice-Hall, 1975.

Marco A Martínez Ramírez and Joshua D Reiss. End-to-end equalization with convolutional neural networks. In *DAFx*, 2018.

Mathieu Ramona and Gaël Richard. A simple and efficient fader estimator for broadcast radio unmixing. In *Proc. Digital Audio Effects (DAFx)*, pages 265–268, 2011.

German Ramos and Jose J Lopez. Filter design method for loudspeaker equalization based on IIR parametric filters. *JAES*, 54(12), 2006.

Marc Rébillat, Romain Hennequin, Etienne Corteel, and Brian FG Katz. Prediction of harmonic distortion generated by electro-dynamic loudspeakers using cascade of hammerstein models. In *Audio Engineering Society Convention 128*. Audio Engineering Society, 2010.

EBU Recommendation. Loudness normalisation and permitted maximum level of audio signals.

Jeffrey S Risberg and Gerald M Shapiro. Digital additive synthesis for computer music. *Journal of the Audio Engineering Society*, 29(12):902–905, 1981.

Tony J Rouphael. *RF and digital signal processing for software-defined radio: a multi-standard multi-mode approach.* Newnes, 2009.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Andy Sarroff and Roth Michaels. Blind arbitrary reverb matching. In *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-2020)*, volume 2, 2020.

Lauri Savioja, Timo Rinne, and Tapio Takala. Simulation of room acoustics with a 3-d finite difference mesh. In *The 1994 International Computer Music Conference, Aarhus, September 12-17, 1994*, pages 463–466. Int. Computer Music Ass. and Danish Inst. of Electroa. Music, 1994.

Roman Scharrer and Michael Vorländer. Blind reverberation time estimation. In *Proceedings of the International Conference on Acoustics, Sydney, Australia*, 2010.

Markus Schedl, Emilia Gómez, Julián Urbano, et al. Music information retrieval: Recent developments and applications. *Foundations and Trends® in Information Retrieval*, 8(2-3): 127–261, 2014.

Michael Schoeffler, Sarah Bartoschek, Fabian-Robert Stöter, Marlene Roess, Susanne Westphal, Bernd Edler, and Jürgen Herre. webmushra—a comprehensive framework for web-based listening tests. *Journal of Open Research Software*, 6(1), 2018.

Johan Schoukens and Lennart Ljung. Nonlinear system identification: A user-oriented road map. *IEEE Control Systems Magazine*, 39(6):28–99, 2019.

Manfred R Schroeder and Benjamin F Logan. " colorless" artificial reverberation. *IRE Transactions on Audio*, (6):209–214, 1961.

Diemo Schwarz and Dominique Fourer. Unmixdb: A dataset for dj-mix information retrieval. In *19th International Symposium on Music Information Retrieval (ISMIR)*, 2018.

Diemo Schwarz and Dominique Fourer. Methods and datasets for dj-mix reverse engineering. In *Perception, Representations, Image, Sound, Music: 14th International Symposium, CMMR 2019, Marseille, France, October 14–18, 2019, Revised Selected Papers 14*, pages 31–47. Springer, 2021.

Ivan W Selesnick and C Sidney Burrus. Generalized digital butterworth filter design. *IEEE Trans. Signal Process.*, 46(6), 1998.

Mike Senior. *Mixing secrets for the small studio.* Taylor & Francis, 2011.

Siyuan Shan, Lamtharn Hantrakul, Jitong Chen, Matt Avent, and David Trevelyan. Differentiable wavetable synthesis. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4598–4602. IEEE, 2022.

Di Sheng and György Fazekas. Automatic control of the dynamic range compressor using a regression model and a reference sound. In *Proceedings of the 20th International Conference on Digital Audio Effects (DAFx-17)*, pages 160–167, 2017.

Larry A. Shepp and Robert J. Vanderbei. The complex zeros of random polynomials. *Trans. Amer. Math. Soc.*, 347(11), 1995. ISSN 0002-9947. doi: 10.2307/2155041.

Julius O. Smith. *Spectral Audio Signal Processing.* `http:`http://ccrma.stanford.edu/ jos/sasp/`//-ccrma.stanford.edu/~jos/sasp/`, accessed 03 November, 2022. online book, 2011 edition.

Reinhard Sonnleitner, Andreas Arzt, and Gerhard Widmer. Landmark-based audio fingerprinting for dj mix monitoring. In *ISMIR*, pages 185–191, 2016.

Michael Stein, Jakob Abeßer, Christian Dittmar, and Gerald Schuller. Automatic detection of audio effects in guitar and bass recordings. In *Audio Engineering Society Convention 128.* Audio Engineering Society, 2010.

Christian J Steinmetz. Learning to mix with neural audio effects in the waveform domain. *MS thesis*, 2020.

Christian J Steinmetz and Joshua D Reiss. Efficient neural networks for real-time modeling of analog dynamic range compression. In *Audio Engineering Society Convention 151.* Audio Engineering Society, 2022.

Christian J Steinmetz, Jordi Pons, Santiago Pascual, and Joan Serrà. Automatic multitrack mixing with a differentiable mixing console of neural audio effects. *arXiv preprint arXiv:2010.10291*, 2020.

Christian J. Steinmetz, Nicholas J. Bryan, and Joshua D. Reiss. Style transfer of audio effects with differentiable signal processing. 2022.

Petre Stoica and Torsten Soderstrom. The steiglitz-mcbride identification algorithm revisited–convergence analysis and accuracy aspects. *IEEE Trans. Automat. Contr*, 26(3), 1981.

Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, pages 334–340, 2018.

Toshiki Tajima and Kazuhiko Kawahara. A case study of cultural influences on mixing preference—targeting japanese acoustic major students. In *Audio Engineering Society Convention 147*. Audio Engineering Society, 2019.

Terence Tao and Van Vu. Random matrices: universality of ESDs and the circular law. *Ann. Probab.*, 38(5), 2010. ISSN 0091-1798. doi: 10.1214/10-AOP534.

Terence Tao and Van Vu. Random matrices: universality of local spectral statistics of non-Hermitian matrices. *Ann. Probab.*, 43(2), 2015a. ISSN 0091-1798. doi: 10.1214/13-AOP876.

Terence Tao and Van Vu. Local universality of zeroes of random polynomials. *Int. Math. Res. Not. IMRN*, (13), 2015b. doi: 10.1093/imrn/rnu084.

Joseph Turian and Max Henry. I'm sorry for your loss: Spectrally-based audio distances are bad at pitch. *arXiv preprint arXiv:2012.04572*, 2020.

George Tzanetakis, Randy Jones, and Kirk McNally. Stereo panning features for classifying recording production style. In *ISMIR*, pages 441–444. Citeseer, 2007.

Cyrus Vahidi, Han Han, Changhong Wang, Mathieu Lagrange, György Fazekas, and Vincent Lostanlen. Mesostructures: Beyond spectrogram loss in differentiable time-frequency analysis. *arXiv preprint arXiv:2301.10183*, 2023.

Vesa Välimäki and Joshua D Reiss. All about audio equalization: Solutions and frontiers. *Applied Sciences*, 6(5):129, 2016.

Vesa Valimaki, Julian D Parker, Lauri Savioja, Julius O Smith, and Jonathan S Abel. Fifty years of artificial reverberation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(5):1421–1448, 2012.

Vesa Välimäki, Julian Parker, Lauri Savioja, Julius O Smith, and Jonathan Abel. More than 50 years of artificial reverberation. In *Audio engineering society conference: 60th international conference: dreams (dereverberation and reverberation of audio, music, and speech)*. Audio Engineering Society, 2016.

Tara Vanhatalo, Pierrick Legrand, Myriam Desainte-Catherine, Pierre Hanna, Antoine Brusco, Guillaume Pille, and Yann Bayle. A review of neural network-based emulation of guitar amplifiers. *Applied Sciences*, 12(12):5894, 2022.

Van Vu. Recent progress in combinatorial random matrix theory. *arXiv:2005.02797*, 2020.

Vesa Välimäki and Joshua D. Reiss. All about audio equalization: Solutions and frontiers. *Applied Sciences*, 6(5), 2016. ISSN 2076-3417. doi: 10.3390/app6050129. URL `https://www.mdpi.com/2076-3417/6/5/129`.

Vesa Välimäki and Jussi Rämö. Neurally controlled graphic equalizer. *IEEE/ACM TALSP.*, 27(12), 2019. doi: 10.1109/TASLP.2019.2935809.

Wil M Wagenaars, Adrianus J Houtsma, and Ruud A van Lieshout. Subjective evaluation of dynamic compression in music. *Journal of the Audio Engineering Society*, 34(1/2):10–18, 1986.

Xin Wang, Shinji Takaki, and Junichi Yamagishi. Neural source-filter waveform models for statistical parametric speech synthesis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:402–415, 2020. doi: 10.1109/TASLP.2019.2956145.

Kurt James Werner, Julius O Smith, and Jonathan S Abel. Wave digital filter adaptors for arbitrary topologies and multiport linear elements. In *Proc. 18th Int. Conf. Digital Audio Effects*, pages 379–386, 2015.

Kurt James Werner, Alberto Bernardini, Julius O Smith, and Augusto Sarti. Modeling circuits with arbitrary topologies and active linear multiports using wave digital filters. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(12):4233–4246, 2018.

Paul White. Automation for the people. *Sound on Sound*, 23(12), 2008.

Alex Wilson and Bruno Fazenda. Categorisation of distortion profiles in relation to audio quality. In *DAFx*, 2014.

Alex Wilson and Bruno Fazenda. 101 mixes: A statistical analysis of mix-variation in a dataset of multi-track music mixes. In *Audio Engineering Society Convention 139*. Audio Engineering Society, 2015.

Alex Wilson and Bruno Fazenda. Variation in multitrack mixes: analysis of low-level audio signal features. *Journal of the Audio Engineering Society*, 64(7/8):466–473, 2016.

Alec Wright, Vesa Välimäki, et al. Grey-box modelling of dynamic range compression. In *Proc. Int. Conf. Digital Audio Effects (DAFX), Vienna, Austria*, pages 304–311, 2022.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

David T Yeh, Jonathan Abel, and Julius O Smith. Simulation of the diode limiter in guitar distortion circuits by numerical solution of ordinary differential equations. *Proceedings of the Digital Audio Effects (DAFx'07)*, pages 197–204, 2007.

David T Yeh, Jonathan S Abel, and Julius O Smith. Automated physical modeling of nonlinear audio circuits for real-time audio effects—part i: Theoretical development. *IEEE transactions on audio, speech, and language processing*, 18(4):728–737, 2009.

Poonna Yospanya, Sorawat Chivapreecha, and Thitaphan Jongsataporn. A deep learning approach to digital filter parameter estimation based on amplitude responses. In *KST*, 2021. doi: 10.1109/KST51265.2021.9415855.